Hierarchical Motion Planning for Autonomous Aerial and Terrestrial Vehicles

A Thesis
Presented to
The Academic Faculty

by

Raghvendra V. Cowlagi

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
Guggenheim School of Aerospace Engineering

Georgia Institute of Technology
August 2011

Hierarchical Motion Planning for Autonomous Aerial and Terrestrial Vehicles

Approved by:

Dr. Panagiotis Tsiotras, Advisor
Guggenheim School of Aerospace Engineering
Georgia Institute of Technology

Dr. Eric Feron
Guggenheim School of Aerospace Engineering
Georgia Institute of Technology

Dr. Mark Costello
Guggenheim School of Aerospace Engineering
Georgia Institute of Technology

Dr. Mike Stilman
College of Computing
Georgia Institute of Technology

Dr. Magnus Egerstedt
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: April 28, 2011

To my parents

Acknowledgements

I wish to thank my advisor, Dr. Panagiotis Tsiotras for his valuable guidance over the course of my doctoral program. Panos' style of advising encourages independent, open-ended thinking; it might leave the new student a bit clueless and frustrated (as I can testify), but in the long run, it is the most conducive for achieving the celebrated goal of doctoral programs of turning the student into the advisor's peer by the end of the program. Working with Panos, I learnt to appreciate the balance between practicality and mathematical rigor: the importance and the skill of grounding in sound mathematical analysis techniques that "work," and of not flooding the engineering literature with cute mathematical results of no practical value. Also, whereas he advises – directly or otherwise – original thinking, in the future I will unabashedly attempt to mimic his tendency of frequently stepping outside his domain of expertise to learn/research diverse topics.

I wish to thank the members of my committee Dr. Mark Costello, Dr. Magnus Egerstedt, Dr. Eric Feron, and Dr. Mike Stilman for their insightful recommendations on my thesis. In particular, Mike's class on Robot Intelligence (fiendishly abbreviated "RIP") was very helpful in understanding and appreciating the problems and methods in motion planning, and more importantly, the context of motion planning in robot intelligence in general. Mike's (and Dr. Egerstedt's) recommendations on comparing my work with a key existing work went a long way in strengthening my thesis.

Attending a well-respected academic institute such as Georgia Tech has its perks in terms of the quality of the various courses offered. Some of the courses I took stood heads and shoulders above the rest: in particular, I would like to express my sincere gratitude towards Dr. Wassim Haddad (for his courses on nonlinear control

and robust control), Dr. Olivier Bauchau (for his course on classical dynamics, which is perhaps the best engineering course I have attended in my entire academic career), and Dr. Mohammad Ghomi (for his insightful course on differential geometry).

I have enjoyed the good fortune of working, at different points in time, with two researchers of the highest quality, and the interactions with whom have gone a long way in shaping my thought process. First, I wish to thank Dr. Sanjay Bhat, with whom I developed an appreciation of the place of mathematics in control theory. Second, I wish to thank Dr. Joseph Saleh, whose lucid thought process is awe-inspiring. Joe's insights into the critical appreciation of technical literature, and into technical writing were/are priceless.

The other perk of attending a well-respected institute such as Georgia Tech is the company. Stathis Bakolas is among the smartest people I know, and over four years of conversations with him on topics ranging from football to control theory to life in academia have been both entertaining and enriching. Girish Chowdhary has been a great friend, and discussions with him on myriad topics have been just as enriching; his wife Rakshita has always been the perfect hostess. I have had the good fortune of interacting with many more smart individuals: Narayanan TV, Anusha V, Claus "Question" Christmann, Maxime Gariel, So Young Kim, Behnood Gholami, Konstantyn Volyansky, Atri Dutta, Sachin Jain, Stathis Velenis, Thomas Jung, Dan Kuehme, Imon Chakraborty, Oktay Arslan, Tim Wang, and Yiming Zhao, to name a few. Also, special thanks to Cindy Pendley for her ample and kind assistance in navigating past academic red tape.

Not to leave out close friends outside Georgia Tech, I must mention Ninad Pradhan, Chetan Danait, Rohit Pradhan, Salil Wadhavkar, Rahul Saxena, Chinmay Belthangady, and Chetan Rathod, all of whom left indelible impressions during my interactions with them in my impressionable (undergraduate) days.

I have been blessed with a family whom I cannot begin to thank in words; so I

will not try. I would not have come as far as writing an "Acknowledgements" page in a doctoral thesis without my parents, my brother Ashish, and my sister Malavika. My conversations on control problems, and life in general, with my brother-in-law Shaunak Bopardikar, and on sundry mathematical topics with my father-in-law have been especially enriching.

And finally, there is Anushree, which is all I need to say about her.

Table of Contents

List of Tables

List of Figures

Summary

Autonomous mobile robots – both aerial and terrestrial vehicles – have gained immense importance due to the broad spectrum of their potential military and civilian applications. One of the indispensable requirements for the autonomy of a mobile vehicle is the vehicle's capability of planning and executing its motion, that is, finding appropriate control inputs for the vehicle such that the resulting vehicle motion satisfies the requirements of the vehicular task. The motion planning and control problem is inherently complex because it involves two disparate sub-problems: (1) satisfaction of the vehicular task requirements, which requires tools from combinatorics and/or formal methods, and (2) design of the vehicle control laws, which requires tools from dynamical systems and control theory.

Accordingly, this problem is usually decomposed and solved over two levels of hierarchy. The higher level, called the *geometric path planning* level, finds a geometric path that satisfies the vehicular task requirements, e.g., obstacle avoidance. The lower level, called the *trajectory planning* level, involves sufficient smoothening of this geometric path followed by a suitable time parametrization to obtain a reference trajectory for the vehicle.

Although simple and efficient, such hierarchical decomposition suffers a serious drawback: the geometric path planner has no information of the kinematical and dynamical constraints of the vehicle. Consequently, the geometric planner may produce paths that the trajectory planner cannot transform into a feasible reference trajectory. Two main ideas appear in the literature to remedy this problem: (a) randomized sampling-based planning, which eliminates the geometric planner altogether by planning in the vehicle state space, and (b) geometric planning supported by feedback

control laws. The former class of methods suffer from a lack of optimality of the resultant trajectory, while the latter class of methods makes a restrictive assumption concerning the vehicle kinematical model.

We propose a hierarchical motion planning framework based on a novel mode of interaction between these two levels of planning. This interaction rests on the solution of a special shortest-path problem on graphs, namely, one using costs defined on multiple edge transitions in the path instead of the usual single edge transition costs. These costs are provided by a local trajectory generation algorithm, which we implement using model predictive control and the concept of effective target sets for simplifying the non-convex constraints involved in the problem. The proposed motion planner ensures "consistency" between the two levels of planning, i.e., a guarantee that the higher level geometric path is always associated with a kinematically and dynamically feasible trajectory.

The main contributions of this thesis are:

1. A *motion planning framework based on history-dependent costs* (H-costs) in cell decomposition graphs for incorporating vehicle dynamical constraints: this framework offers distinct advantages in comparison with the competing approaches of discretization of the state space, of randomized sampling-based motion planning, and of local feedback-based, decoupled hierarchical motion planning,

2. An efficient and flexible *algorithm for finding optimal H-cost paths*,

3. A precise and general *formulation of a local trajectory problem* (the tile motion planning problem) that allows independent development of the discrete planner and the trajectory planner, while maintaining "compatibility" between the two planners,

4. A local trajectory generation algorithm using MPC, and the *application of the*

concept of effective target sets for a significant simplification of the local trajectory generation problem,

5. The *geometric analysis of curvature-bounded traversal* of rectangular channels, leading to less conservative results in comparison with a result reported in the literature, and also to the efficient construction of effective target sets for the solution of the tile motion planning problem,

6. A wavelet-based multi-resolution path planning scheme, and a *proof of completeness* of the proposed scheme: such proofs are altogether absent from other works on multi-resolution path planning,

7. A technique for extracting *all* information about cells – namely, the locations, the sizes, and the associated image intensities – directly from the set of significant detail coefficients considered for path planning at a given iteration, and

8. The extension of the multi-resolution path planning scheme to include vehicle dynamical constraints using the aforementioned history-dependent costs approach.

The future work includes an implementation of the proposed framework involving a discrete planner that solves classical planning problems more general than the single-query path planning problem considered thus far, and involving trajectory generation schemes for realistic vehicle dynamical models such as the bicycle model.

Chapter 1

Introduction and Literature Review

Autonomous terrestrial and aerial vehicles have gained immense importance not only due to the broad spectrum of their potential military and commercial applications, but also due to the concurrent development of sensor technology and embedded systems that enable the realization of true autonomy. Autonomous vehicles may be assigned tasks that are dull and/or repetitive, such as mobile surveillance or cleaning and maintenance; tasks that are dangerous for humans, such as military transportation via hostile territory, large-scale fire fighting, and repair and recovery operations in chemical plants and nuclear reactors; or tasks that are prohibitively expensive for humans to execute, such as the exploration of celestial bodies. Unsurprisingly, various theoretical and practical aspects of the development of autonomous mobile vehicles have been vigorously and extensively researched by the robotics, automatic control, and artificial intelligence communities for over four decades [30, 54, 93, 139]. Nevertheless, whereas fixed-base robotic manipulators are commonplace in the industry, fully autonomous *mobile* robots are yet the realm of state-of-the-art research (Fig. 1.1 depicts two such research vehicles), and several issues in this field of engineering are

(a) Google™'s autonomous car [1]

(b) Carnegie Mellon University's "Boss" [148]

Figure 1.1: State-of-the-art autonomous vehicles.

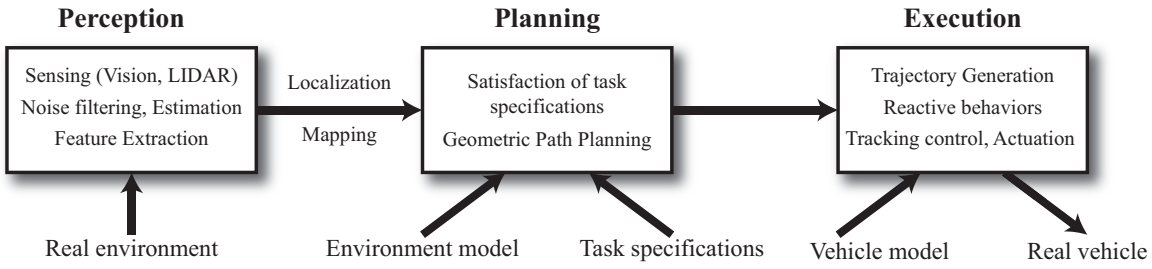


Figure 1.2: The typical "perceive-plan-execute" hierarchical control structure of autonomous vehicles (adapted from [139]).

open topics of research.

Autonomous motion is enabled by the following, equally crucial sub-systems: perception, planning, and control. Accordingly, a "perceive-plan-execute" hierarchy, illustrated in Fig. 1.2, has traditionally been employed for realizing autonomous motion [139]. Owing both to the challenges involved in the development of each of these sub-systems themselves, and to the involvement of different, traditionally disconnected academic communities – perception and planning have traditionally been addressed by the robotics and artificial intelligence communities, whereas low-level motion execution has traditionally been addressed by the automatic control community – the inter-dependence of these sub-systems is yet to be adequately and rigorously addressed [139]. The current motion planning and control literature, for example, indicates a paradigm shift of focus from the traditional single-query ("go from A to B") problems to broader problems involving the generation of provably correct control laws starting from high-level task specifications (cf. [12, 81, 158]), thus blurring the distinction between classical planning and automatic control.

In this thesis, we investigate the problem of *motion planning and control* for autonomous vehicles. Informally, this problem deals with finding appropriate control inputs for an autonomous vehicle such that the vehicle's resulting motion satisfies the requirements of a specified task. This problem is inherently complex because it involves two disparate sub-problems: (1) the satisfaction of the vehicular task requirements, which requires tools from combinatorics and/or formal methods, and (2) the

design of the vehicle control laws, which requires tools from dynamical systems and control theory. To address the complexity of the motion planning and control problem, two distinct approaches have been proposed in the literature: a "top-down" approach that concentrates on planning a geometric path compatible with the task requirements, which is then followed by a low-level controller that executes this geometric path; and a "bottom-up" approach that concentrates on control-driven discretizations of the vehicle motions, which are then appropriately concatenated to meet the vehicular task. A vast majority of the literature in motion planning (cf. [30, 90, 93]) falls into the former category.

A unified view [139] of the two aforementioned approaches involves the decomposition of motion planning and control systems into two sub-systems: (1) a strategic, or "deliberative" sub-system, that deals with global, long-term navigation and path planning and with the satisfaction of the vehicular task requirements, and (2) a tactical, or "reactive" sub-system that deals with short-term, local navigation and obstacle avoidance in the presence of the vehicle's kinematic and dynamic constraints. This decomposition is both conceptually and practically appealing; however, in the absence of guarantees of "compatibility" of the two said sub-systems, the resultant motion of the vehicle may consist of control tactics that are strategically infeasible or unacceptably sub-optimal; or conversely, strategies that call for infeasible or unacceptably sub-optimal control tactics.

In this thesis, we investigate the interaction between these two sub-systems to develop an overall motion planning and control framework which allows independent development of either sub-system, while maintaining a guarantee of "compatibility" between the two sub-systems.

1.1 Terminology and Formal Problem Statement

In this section, we establish the basic terminology and notation to be used throughout this thesis, and we state precisely the motion planning and control problem.

We will use the term *path* to refer to the locus of the continuous motion of a point in the plane, and the term *trajectory*, or synonymously, the term *motion* to refer to a path or a curve parametrized by time. Depending on the context, we will use the term *path* to refer also to a sequence of successively adjacent vertices in a graph.

In this thesis, we consider vehicles moving in the plane. In this context, we use the term *workspace*, or synonymously, the term *environment* to refer to a planar region $\mathcal{W} \subset \mathbb{R}^2$ over which the vehicle moves. The *obstacle space* $\mathcal{O} \subset \mathcal{W}$ is a subset of the workspace. The vehicle's *configuration space* $\mathcal{C} := \mathbb{R}^2 \times \mathbb{S}$ is the set of all vehicle configurations, i.e., its position in the plane and its orientation. Finally, the term *state space* refers to the state space $\mathcal{D}$ of the dynamical model of the vehicle, which includes the configuration space, but may be larger.

We consider a vehicle dynamical model described as follows. Let $(x, y, \theta) \in \mathcal{C}$ denote the position coordinates of the vehicle in a pre-specified Cartesian axis system, and let ψ denote any additional state variables required to describe the state of the vehicle. We assume that $\psi \in \mathcal{M}$, where $\mathcal{M}$ is a n-dimensional smooth manifold. The state of the vehicle is thus described by $\xi := (x, y, \theta, \psi) \in \mathcal{D} = \mathcal{C} \times \mathcal{M}$. Let $U \in \mathbb{R}^m$ denote the set of admissible control values; and for $t > 0$, let $\mathcal{U}_t$ denote the set of piecewise continuous functions defined on the interval $[0, t]$ that take values in U. We assume that the evolution of the vehicle state ξ over a given time interval $[0, t_\mathrm{f}]$ is described by the differential equation $\dot{\xi}(t) = f(\xi(t), u(t))$ for all $t \geqslant [0, t_\mathrm{f}]$, where $u \in \mathcal{U}_{t_\mathrm{f}}$ is an admissible control input, and f is sufficiently smooth to guarantee global existence and uniqueness of solutions. We denote by $\xi(\cdot\,; \xi_0, u)$ the state trajectory that is the unique solution to the preceding differential equation with the initial condition $\xi(0) = \xi_0$.

The motion of an autonomous vehicle is usually required to satisfy a *task*, ranging from the relatively simple (e.g., "go from A to B") to the relatively complex (e.g., "collect and analyze rock samples"). Tasks are typically specified by formulae of predicate or temporal logic over a finite set of objects, and the problem of satisfaction of task specifications is typically formulated as a problem of classical planning or formal verification [12, 130]. Associated typically with the task satisfaction problem is a finite state transition system consisting of a set $\mathcal{T}$ of *task states*, a finite set $\mathcal{A}$ of *actions*, and a *transition function* $\delta : \mathcal{T} \times \mathcal{A} \to \mathcal{T}$ that associates with a state-action pair a new task state.

Of our interest are tasks directly related to the motion of the autonomous vehicle in the workspace $\mathcal{W}$, and the task specification hence necessitates a finite discretization of the workspace. In this thesis, we assume, as is the popular choice [12], that such a discretization of the workspace is achieved via *workspace cell decomposition*, i.e., a partition of the workspace into a finite number of convex, obstacle-free regions called *cells*. In the context of the classical planning problem formulation [130], task states are defined by predicate logic formulae, the task specification consists of a goal state in $\mathcal{T}$ that must be reached via a sequence of actions, and each task state is associated (not necessarily uniquely) with a cell. On the other hand, temporal logic formulae are used to specify constraints on the vehicle's behavior over the period of its operation [81, 158]; the task states then directly correspond to the cells, the transition function captures geometric adjacency relations between the cells, and the task specification consists of constraints on the sequence of cells traversed by the vehicle.

Using the preceding terminology, the general motion planning problem for autonomous vehicles may be formulated as follows.

Problem 1.1 (General Motion Planning)**.** *Given a task specification, an initial task state τ_0, and an initial vehicle state ξ_0, find a sequence $\{a_m\}_{m=1}^{P} \subset \mathcal{A}$ of actions, a*

number $t_{\mathrm{f}} \in \mathbb{R}_+$*, and an admissible control input* $u \in \mathcal{U}_{t_{\mathrm{f}}}$ *such that*

(a) The sequence $\{\tau_n\}_{n=0}^{P}$ *of task states, defined by* $\tau_n := \delta(\tau_{n-1}, a_n)$ *for each* $n = 1, \ldots, P$ *satisfies the task specification, and*

(b) The trajectory $\xi(t; \xi_0, u)$ *enables the vehicle's traversal through the sequence of cells associated with the sequence* $\{\tau_n\}_{n=0}^{P}$ *of task states.*

In this thesis, we first focus on the *point-to-point motion planning* problem, where the vehicle task is to navigate from a pre-specified initial point in the workspace to a pre-specified destination. In Chapter 8, we address the extension to the general motion planning problem of the ideas presented in this thesis for solving the point-to-point problem. For the point-to-point problem, we identify the task space with the set of all cells in the workspace cell decomposition, and we associate each cell with a unique task state. Also, we identify actions with cell transitions. A hybrid[1] formulation of the point-to-point motion planning problem is then stated as follows.

Problem 1.2 (Hybrid Point-to-Point Motion Planning)**.** *Given an initial cell* τ_0*, a goal cell* τ_{f}*, and an initial vehicle state* ξ_0*, find a sequence* $\{a_m\}_{m=1}^{P} \subset \mathcal{A}$ *of cell transitions, a number* $t_{\mathrm{f}} \in \mathbb{R}_+$*, and an admissible control input* $u \in \mathcal{U}_{t_{\mathrm{f}}}$ *such that*

(a) The sequence $\{\tau_n\}_{n=0}^{P}$ *of cells, defined by* $\tau_n := \delta(\tau_{n-1}, a_n)$ *for each* $n = 1, \ldots, P$ *satisfies the specification* $\tau_P = \tau_{\mathrm{f}}$*, and*

(b) The trajectory $\xi(t; \xi_0, u)$ *enables the vehicle's traversal through the sequence of cells* $\{\tau_n\}_{n=0}^{P}$*.*

It is important to note that the point-to-point motion planning problem may be stated, as below, without reference to a discretization of the workspace.

[1]Here the term "hybrid" refers to the involvement of a discrete sub-problem (formulated on the task space) as well as a continuous sub-problem (formulated on the vehicle state space).

Problem 1.3 (Continuous Point-to-Point Motion Planning)**.** *Given an initial vehicle state* ξ_0 *and a terminal state* ξ_{f}*, find a number* $t_{\mathrm{f}} \in \mathbb{R}_+$ *and a control input* $u \in \mathcal{U}_{t_{\mathrm{f}}}$ *such that the trajectory* $\xi(t; \xi_0, u)$ *satisfies*

$$\xi(t; \xi_0, u) \in \mathcal{W} \cap \overline{\mathcal{O}}, \qquad \textit{for all } t \in [0, t_{\mathrm{f}}]$$

and $\xi(t_{\mathrm{f}}; \xi_0, u) = \xi_{\mathrm{f}}$*.*

All of the motion planning problems stated so far refer only to *feasible* solutions. For the simpler, point-to-point problem, we may additionally incorporate in the problem a notion of *optimality*, as follows.

Problem 1.4 (Optimal Point-to-Point Motion Planning)**.** *Let* $\ell : \mathcal{D} \times U \times \mathbb{R}_+ \to \mathbb{R}_+$ *be a pre-specified incremental trajectory cost function. Given an initial vehicle state* ξ_0 *and a terminal state* ξ_{f}*, solve*

$$\min_{u \in \mathcal{U}_{t_{\mathrm{f}}}} \int_0^{t_{\mathrm{f}}} \ell(\xi(t; \xi_0, u), u, t) \, \mathrm{d}t,$$

subject to the constraint

$$\xi(t; \xi_0, u) \in \mathcal{W} \cap \overline{\mathcal{O}}, \qquad \textit{for all } t \in [0, t_{\mathrm{f}}]\,,$$

where t_{f} *is free, and* $\xi(t_{\mathrm{f}}; \xi_0, u) = \xi_{\mathrm{f}}$*.*

The motion planning problem and several underlying fundamental issues have been addressed in the literature pertaining to at least three different research disciplines: artificial intelligence, robotics, and automatic control. In the next two sections, we survey the literature directly related to the motion planning problem, and we review some of the fundamental theoretical background necessary to better understand and solve this problem.

1.2 Theoretical Background and Literature Review

The following fields of study form the theoretical underpinnings of solutions to the motion planning problem: *optimal control* theory addresses the problem of trajectory

generation for point-to-point motion under differential constraints; *classical planning* theory in artificial intelligence addresses the generation of a finite sequence of actions to satisfy task specifications; *hierarchical systems* theory analyzes the interactions between sub-systems in a hierarchical system for ensuring "harmonious" operation of the overall system; and the emerging field of *hybrid systems and control* theory addresses the analysis and control of systems involving both discrete and continuous state variables.

In light of the multi-disciplinary theoretical background required for addressing the motion planning problem in its full generality, we provide in Appendix A brief reviews of these fields of study, and therein we refer the reader to appropriate texts for further details. We also review briefly in Appendix A the theory of the discrete wavelet transform, which is a powerful mathematical tool frequently used in signal processing, and by consequence, in autonomous navigation systems.

Based on the aforementioned theoretical developments, several different ideas have been considered and developed for path- and motion planning for autonomous mobile vehicles and for robotic manipulators. Figure 1.3 illustrates schematically the evolution of these ideas; in what follows, we provide a brief summary of the path- and motion planning literature. The reader interested is referred to the works of Latombe [90], Hwang and Ahuja [60], Choset *et al* [30], and LaValle [93] for comprehensive surveys; the works of Belta *et al* [12] and Frazzoli [46] provide surveys of control theoretic perspectives on this subject.

1.2.1 Geometric Path Planning

Geometric path planning methods are computationally efficient techniques that attempt typically to find obstacle-free paths in the workspace. Three broad ideas have emerged in geometric path planning [30, 90]: roadmaps, and cell decompositions, and artificial potential fields. Two of these ideas – roadmaps and cell decompositions –

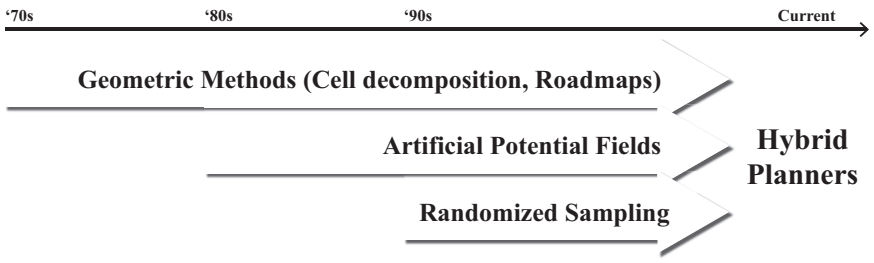


Figure 1.3: The evolution of the path- and motion planning literature.

involve discretization of the workspace to transform the path planning problem into a graph search problem. Geometric path planning methods do not typically consider the vehicle's kinematic and dynamic constraints.

1.2.1.1 Roadmaps

Conceptually, roadmaps [90, Ch. 4], [30, Ch. 5] are graphs whose vertices are locations in the free workspace and whose edges are paths or curves lying completely in the free space. The simplest and the earliest of roadmaps is the visibility graph [103, 126], in which two vertices are connected if they are in line of sight of each other (see Fig. 1.4). Extensions to the basic visibility graph-based methods include works addressing reductions in the number of edges of the graph [38,57,66], works addressing curved obstacles [102], and involving 3D path planning [68]. Path planning methods based on Voronoi partitions of the environment [38, Ch. 7], [17, 29, 70, 87, 127] form another class of important roadmap methods.

1.2.1.2 Artificial potential fields

Path planning based on artificial potential fields was introduced by Khatib [76] and later generalized to the so-called navigation functions by Rimon and Koditschek [124]. Conceptually, an artificial potential field or a navigation function is a scalar field that has a global minimum at the desired destination point; path planning from any point in the environment is then performed by following the direction of steepest descent. The single most important problem involved in the construction of artificial

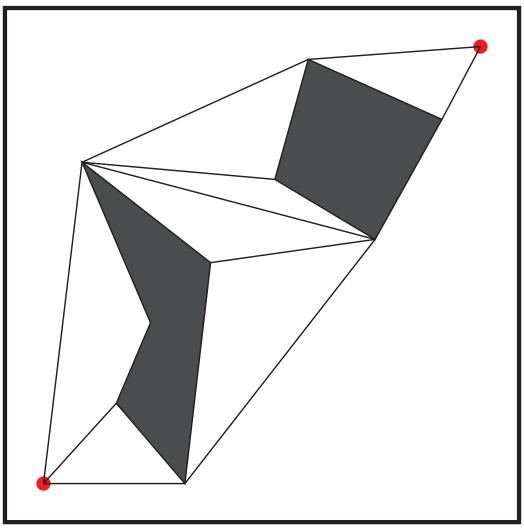
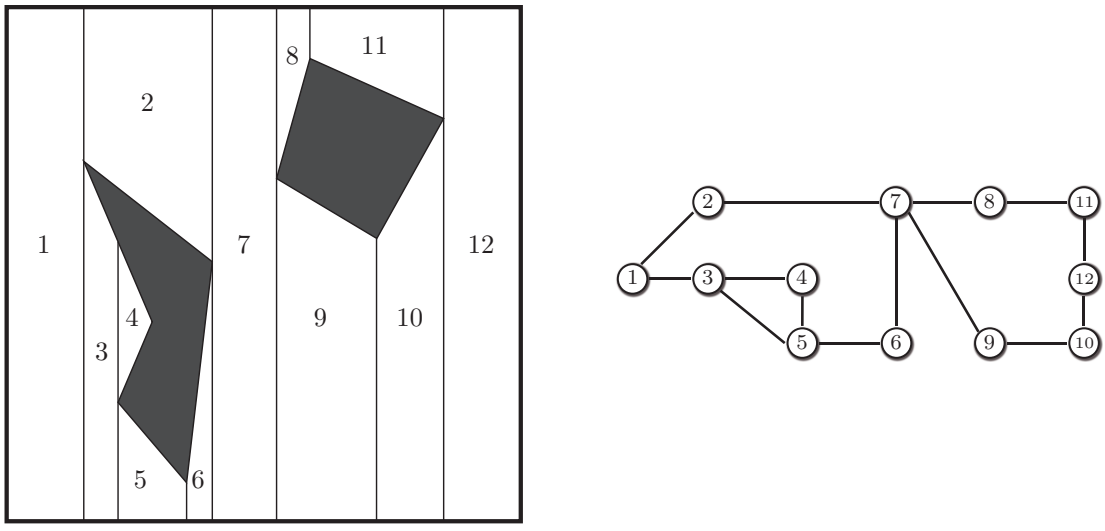


Figure 1.4: Example of a visibility roadmap.

potential fields is the presence of unwanted local minima that may mislead a vehicle following the direction of the negative local gradient. References [9, 48, 61, 78, 79, 154] are examples of applications and extensions of the potential function and navigation function methods to path planning.

1.2.1.3 Cell decompositions

Geometric path planning based on cell decompositions, introduced by Brooks and Lozano-Pérez [22], involves partitioning the environment into convex, obstacle-free regions called *cells*. A graph $\mathcal{G}$ is then associated with the cell decomposition, where each cell is represented by a vertex in the graph, and geometric adjacency of the cells is represented by edges. A path in this graph from a pre-specified initial cell to a pre-specified goal cell then corresponds to a sequence of cells from the initial cell to the goal cell (see Fig. 1.5). *Exact* cell decompositions are partitions of the environment in which the union of all cells is exactly equal to the free workspace. On the other hand, *approximate* cell decompositions use convenient geometric shapes such as squares or rectangles to partition the environment. Because obstacles can be of arbitrary shape, cells in approximate cell decompositions can be *mixed*, i.e., contain both free space



(a) Exact cell decomposition using linear sweep.

(b) Associated topological graph.

Figure 1.5: Illustration of workspace cell decomposition.

and obstacle space.

Triangular and trapezoidal decompositions [38], [90, Ch.4], [30, Ch.6], are the most widely used exact cell decomposition techniques, whereas the quadtree method [71,113,131], which employs recursive decompositions of MIXED cells into four square subcells (children) until all cells are obstacle-free, is the most extensively used approximate cell decomposition technique.

Configuration space cell decompositions have been applied to solve the piano mover's problem [133, 134], and also to path planning for robotic manipulators [55]. Extensions to the standard cell decomposition technique include efficient techniques of determining if a region in the workspace is obstacle-free [67], and efforts for finding shortest Euclidean path between the initial point and the goal [26].

Path planning schemes using *multi-resolution cell decompositions* have been discussed in the literature. Multi-resolution path planning involves representing the vehicle's environment with different levels of accuracy to construct an overall representation that allows for efficient online path planning. For example, the quadtree method [71, 113, 131], generates a planar cell decomposition consisting of small cell

sizes that accurately capture obstacle boundaries, and larger cell sizes that efficiently represent large areas in free space. The quadtree method employs recursive decompositions of MIXED cells into four square subcells (children) until all cells are either FREE or FULL, and it is one of the most extensively used multi-resolution cell decomposition technique. Other path planning schemes using multi-resolution cell decompositions have been previously proposed in [11]; in [59], where triangular cells are used in a more efficient cell decomposition; in [119], which discusses a receding horizon path planning scheme using multi-resolution estimates of object locations; in [77], which presents a multi-resolution potential field approach; and in [152], which discusses a hierarchy of imaginary spheres encapsulating the robot for collision avoidance.

The wavelet transform has frequently been applied in multi-resolution path planning algorithms. For instance, Pai and Reissell [114] present an algorithm for path planning over a rough terrain. They iteratively refine the path based on successively finer approximations of the terrain elevation map. At each iteration, the wavelet transform coefficients of the elevation map are used to compute the approximation errors, which are then included in the cost function that should be minimized by the optimal path. Sinopoli *et al* [140] describe a similar approach for vision-based path planning for autonomous UAVs. Carrioli [24] describes the use of the Haar wavelet for reducing computations, while manipulating images representing the environment. In a slightly different application, Narayanswami and Pang [111] describe a path planning algorithm for NC machining that uses successively finer approximations to the required contour as the cutting tool approaches the contour.

1.2.2 Nonholonomic Path Planning

A particular class of kinematical models, namely, *nonholonomic models*, present special challenges in path planning due to the fact that nonholonomicvelocity constraints cannot be integrated to obtain position constraints (which could then be treated as

obstacles). The most widely studied nonholonomic models of mobile vehicles are the Dubins car model (a car-like vehicle that can only move forwards), and the Reeds-Shepp car model (a car-like model that can move both forwards and backwards). Nonholonomic constraints in both these models imply that the curvature of the geometric paths that can be feasibly followed by the vehicle is finitely upper-bounded, i.e., instantaneous changes in the tangent to the path are not permitted. Consequently, the results of the previously discussed geometric path planners do not suffice for nonholonomic vehicles.

Curvature-bounded path planning in the absence of obstacles has been studied in the seminal works of Dubins [40] (for a vehicle that can move only forwards) and Reeds and Shepp [122] (for a vehicle that can move forwards and backwards). The primary results of these papers are that, in the absence of workspace obstacles, the shortest curvature-bounded paths belong to particular families of paths consisting of concatenations of straight lines segments and arcs of circles of unit radius[2]. These families of paths are usually referred to as *Dubins paths* and *Reeds-Shepp paths* respectively. Boissonnat *et al* [19] and Sussmann and Tang [144] reproduce, respectively, the results of Dubins and Reeds and Shepp using optimal control theory, while Bui *et al* [23] and Souères *et al* [141] present, respectively, synthesis algorithms for those two problems.

The problem of planning curvature-bounded paths is significantly more involved in the presence of workspace obstacles. Reif and Wang [123] show that the problem of constructing shortest curvature- bounded paths in a polygonal environment with polygonal obstacles or holes is NP-hard. A crucial and fundamental difference between the Dubins and Reeds-Shepp models is that the latter model is completely

[2]Because the Reeds-Shepp car can move backwards, cusps are allowed in the path, whereas paths for the Dubins car cannot involve cusps.

controllable[3] even in the presence of obstacles [10], while the former model is completely controllable only in the absence of obstacles. Consequently, in the case of path planning for the Dubins car model, it becomes necessary to first verify if a kinematically feasible path *exists* between the initial configuration and the goal configuration.

Fortune and Wilfong [44] present an algorithm for deciding whether an obstacle-free, curvature-bounded path without cusps exists in a polygonal environment with polygonal obstacles. However, by the authors' own admission, their algorithm is very complicated, and does not *find* a path (it only guarantees existence). References [2, 7, 18, 63] present algorithms for efficiently finding approximations to shortest curvature-bounded paths in polygonal environments. The common theme in those references is to appropriately choose a finite set of configurations (position and orientation) in the environment, and connecting these configurations by Dubins paths, and then perform a graph search using the length of the Dubins paths as edge costs. References [10, 89, 92, 109, 110, 160] deal with the problem of planning curvature-bounded paths for the Reeds-Shepp car. Laumond *et al* [91] provide a survey of motion planning and control techniques for both the Dubins and Reeds-Shepp vehicle models.

References [8, 33, 45, 132, 159] discuss a further extension of non-holonomic path planning, namely, *continuous curvature path planning*, where the *derivative* of the tangent is required to be continuous. Continuous curvature paths are important because vehicles with bounded control inputs cannot perfectly track paths with instantaneous changes in the derivative of the tangent to the path.

1.2.3 Randomized Sampling-based Motion Planning

Randomized sampling-based algorithms were first developed for path planning in high dimensional configuration spaces associated with robotic manipulators and humanoid

[3]Complete controllability means that a kinematically feasible path exists between any two configurations in the free space.

robots. Conceptually, these algorithms rely on randomly selected configurations (instead of configurations on a grid) to discretize the configuration space.

Probabilistic roadmap planners [73, 74, 153], [30, Ch. 7] use uniformly distributed random sampling of the free space to establish the vertices of a roadmap. A local planning and collision detection algorithm is used to determine edges between vertices. Rapidly-exploring Random Trees (RRT), invented by LaValle and Kuffner [95], are data structures that use efficient sampling strategies to quickly explore high dimensional spaces. LaValle and Kuffner [94], Hsu and Latombe [56] and Frazzoli *et al* [47] discuss the applications of RRTs to motion planning in the *state space* with dynamical constraints. The efficiency of randomized sampling-based algorithms is dependent on the fast expansion of the data structure (such as the RRT) used to represent the configuration space/state space, which is typically achieved via linear interpolation between a configuration/state already present in the tree and a randomly explored new configuration/state.

Whereas the preceding strategy suffices for planning in configuration spaces with only *geometric* constraints, linear interpolation between two states does not, in general, correspond to an admissible state trajectory in the presence of *differential* constraints, such as those involved in the equations of motion of a controlled dynamical system. The expansion of the tree is then achieved by integrating forward with a randomly selected input [94] the dynamical constraint equations, using as initial condition a state in the tree; or by connecting a known state in the tree with a new state by solving a numerical optimal control problem. Note that he latter approach is tantamount to solving the *original* point-to-point motion planning problem via numerical optimal control techniques.

A related issue involved with the expansion of RRTs in state spaces in the presence of differential constraints is the appropriate selection of the nearest neighbor, i.e., the state in the tree that is "nearest" to a randomly explored new state. Whereas there

exist efficient algorithms (cf. [94, 130]) for find the "nearest" neighbor in the context of Euclidean distances, finding the nearest neighbor in terms of other metrics (such as time of traversal) is not straightforward.

1.2.4 Hierarchical Motion Planning

The literature on path- and motion planning dealing with kinematical/dynamical constraints is dominated by hierarchical approaches [27, 33, 36, 92, 108, 128, 136, 164] – representatives of the aforementioned "top-down" approach to motion planning – that decompose the problem into a high-level, discrete, geometric planning level and a low-level, continuous, trajectory planning level. In the context of low-level trajectory planning, Refs. [96, 135, 137, 151] discuss time-optimal trajectory planning along pre-specified geometric paths for specific vehicle dynamics.

On the other hand, Donald *et al* [39] introduce the "bottom-up" approach based on constructing a discrete representation of the vehicle's motion (as opposed to that of the environment) – the authors of [39] also introduce the term *kinodynamic* motion planning to refer to the problem of planning with kinematical and dynamical constraints. Another major contribution to the "bottom-up" approach to motion planning is Frazzoli's work on the maneuver automaton [46].

Motion planning methods using cell decomposition-based *path planning coupled with feedback control laws* comprise another important set of contributions in the literature that address vehicle's kinematical and dynamical constraints [13,32,99–101]. These works provide methods to generate reference vector fields within cells that guarantee transition through any given sequence of cells without intersecting any other cell. Fainekos *et al* [42, 43] and Kloetzer and Belta [82] (in the context of linear control) also use this idea to develop solutions that are guaranteed to satisfy both aspects of motion planning and control: vehicular task specifications that are expressed as temporal logic formulae, as well as kinematical and dynamical constraints

that expressed as differential equations. The common idea used in all of the preceding references is to make the geometric planner *independent* of the vehicle kinematics an dynamics; i.e., to ensure that *every* sequence of cell transitions is possible from *every* initial vehicle state.

1.3 Thesis Overview and Statement of Contributions

We make the following broad observations concerning the existing literature related to path- and motion planning: computationally efficient techniques that address path optimality (geometric path planners) do not typically consider kinematic/dynamic constraints; computationally efficient techniques that address kinematic/dynamic constraints (randomized sampling-based algorithms) do not typically address optimality; and finally, techniques that address both optimality and kinematic/dynamic constraints are not computationally efficient enough for real-time, online implementations. In this thesis, we address the problem of developing the ideal, computationally efficient point-to-point motion planning technique that addresses both optimality and kinematic/dynamic constraints, with the aim of extending easily the proposed technique to the general motion planning problem.

We address the hybrid point-to-point motion planning problem (Problem 1.2), and we consider for its solution the hierarchical separation of the solution into a (discrete) path planning algorithm based on cell decompositions and a trajectory generation algorithm. This hierarchical separation is advantageous because it relates closely to the general motion planning problem (Problem 1.1), and the techniques developed for the solution of Problem 1.2 may be extended for the solution of Problem 1.1.

The primary drawback of hierarchically separated solutions of the point-to-point motion planning problem is that the sequence of cells resulting from the discrete path planner may not be traversable due to the vehicle's kinematic and dynamic constraints. Furthermore, because nonholonomic kinematic constraints cannot be

integrated to obtain position constraints, it is fundamentally impossible to encode these constraints in the cell decomposition graph.

In this thesis, we propose a framework for motion planning which enables an interaction between the discrete path planner and the trajectory planner to guarantee compatibility between the two planners. This interaction is enabled by defining costs on multiple edge transitions of the cell decomposition graph, instead of the usual single-edge transition costs. We discuss in detail a specific implementation of this framework, where the discrete path planner is implemented using multi-resolution cell decompositions, and the trajectory planner is implemented using model predictive control.

In what follows, we present an overview of the various facets of the proposed motion planning framework.

1.3.1 History-dependent Costs in Path Planning

It has been noted in several previous works [88, 125, 157] that single-edge transition costs cannot capture adequately the vehicle's kinematic and dynamic constraints. In this thesis, we formalize the notion of associating costs to multiple edge transitions in the cell decomposition graph. To this end, we introduce the so-called *H-cost shortest path problem* in Chapter 2, which is the problem of finding optimal paths in graphs where transition costs are defined on $H + 1$ successive edges. We observe that the H-cost shortest path problem may be transformed to an equivalent standard shortest path problem on a different graph, which we call the *lifted graph*. The lifted graph is the primary conceptual tool enabling the interaction between the discrete path planner and the trajectory planner.

In Section 2.2, we propose an efficient and flexible algorithm for solving the H-cost shortest path problem. The proposed algorithm executes significantly faster than the solution of the equivalent standard shortest path problem on the lifted

graph. Furthermore, the proposed algorithm includes a user-specified parameter that allows further reductions in the execution time at the expense of the optimality of the resultant path. In Section 2.3, we provide numerical simulation data to corroborate these claims, for both the optimal and sub-optimal implementations.

In Chapter 3, we develop a motion planning framework based on the H-cost shortest path problem. This framework enables the interaction between the two levels of planning via a special local trajectory generation problem, which we call the *tile motion planning problem*. We provide a precise and general formulation of the tile motion planning problem, and we discuss the solution of an H-cost shortest path problem where the H-costs are provided by the tile motion planner.

The novelty of the proposed motion planning framework is that it provides a clear distinction between the "intelligence" and the "control" aspects involved in motion planning. More precisely, the tile motion planning problem provides a specific interface through which control theoretic ideas may be applied to motion planning *without interfering* with the higher-level planning algorithm; on the other hand, the lifted graph provides a specific discrete mathematical structure on which high level task specifications may be formulated and solved, also without interfering with the tile motion planner.

To further distinguish the proposed approach to motion planning from the existing literature, we note three major approaches in the motion planning literature that are also intended for incorporating vehicle dynamics in motion planning. We state the perspectives of these competing approaches and we emphasize, in comparison to each, the novelty of the proposed approach as follows.

1. **History-dependent costs are not required if discretizations of the state space, instead of the workspace, are considered.** This approach suffers from the lack of scalability to higher-dimensional state spaces. Discretization of the workspace is advisable because the obstacle space is contained within

the workspace, and because this discretization is independent of the vehicle dynamical model (in particular, it is independent of the dimension of the state space). To enable the incorporation of vehicle dynamics in motion planning *independently of the dimension of the state space*, we propose a framework that relies on workspace discretization.

2. **Discretization (of either the workspace or the state space) is not necessary; randomized sampling-based algorithms can plan efficiently even in high dimensional state spaces.** The efficiency of randomized sampling-based algorithms arises mainly from the fast expansion of data structures such as the RRT, which in turn is enabled by fast linear interpolation between a known state and a new state. However, in the presence of differential state constraints (such as vehicle dynamical constraints) and input constraints, linear interpolation between two states does *not*, in general, correspond to a feasible state trajectory. Consequently, the efficiency of randomized sampling-based algorithms deteriorates significantly [118] when vehicle dynamical constraints are considered.

   More importantly, randomized sampling-based planners typically do not address the optimality of the resultant path (with recent notable exceptions [64, 72]). The standard notion of *a posteriori* path "smoothing," which involves the removal of redundant intermediate states in the path, does not apply easily when dynamical constraints are considered, for reasons discussed in the previous paragraph.

   In comparison with randomized sampling-based planners, our proposed approach of enabling by a low-level trajectory planner the discrete search for *optimal* paths results in trajectories with significantly lower costs, at comparable computational efficiency. In Section 7.2, we corroborate this claim with

numerical simulation results.

Finally, a discrete representation of the workspace is typically *required* for stating high-level task specifications [12, 13] more complex than the point-to-point navigation task.

3. **Local feedback laws within cells can decouple the geometric path planning from the trajectory generation.** An underlying assumption in the feedback-based motion planning approach (cf. [13, 32]) is the complete controllability of the vehicle dynamical model in the presence of obstacles, i.e., the assumption that there exists a feasible, obstacle-free trajectory from *every* initial state to *every* goal state. Stated differently, an underlying assumption involved in the design of local feedback laws that enable transitions among cells is the *existence* of such feedback laws. However, such feedback laws do not exist for important vehicle models such as the Dubins car model and the aircraft navigational model. In this thesis, we do *not* assume complete controllability in the presence of obstacles, and we present numerical simulation results for vehicle models which violate this assumption.

1.3.2 Local Trajectory Generation via Model Predictive Control

As previously mentioned, the tile motion planning problem provides a specific interface through which control theoretic ideas may be applied to motion planning. We discuss in Chapter 4 a specific example of such an application of control theory, namely, an implementation of the tile motion planner TilePlan based on the well-known model predictive control (MPC) paradigm. The tile motion planning problem, discussed in detail in Section 3.2, involves trajectory generation for the traversal of a finite sequence of cells. This problem is difficult mainly because the constraint of containment of the trajectory within the said sequence of cells is, in general, a non-convex constraint.

As such, the tile motion planning problem may be solved without further simplification using nonlinear programming (NLP) or MILP techniques in the MPC implementation. This approach, however, may require significant computational resources, especially in the context of real-time, online implementations, and numerical NLP techniques are not, in general, be guaranteed to converge successfully.

We discuss an approach of simplifying the non-convex constraint of containment within a sequence of cells to the convex constraint of containment within a single cell using the concept of the so-called *effective target sets*, first introduced by Bertsekas and Rhodes [16]. Whereas the application of effective target sets simplify the implementation of TilePlan, the construction of these sets is challenging. We address the construction of these sets by first observing that the vehicle's (nonholonomic) kinematic and dynamic constraints impose an upper bound on the local curvature of the geometric paths that can feasibly be traversed by the vehicle.

In light of this observation, we discuss in Chapter 5 a purely geometric scheme for efficiently constructing the effective target *configuration* sets, i.e., the intersections of the effective target sets with the configuration space $\mathcal{C}$. Interestingly, the construction of the effective target configuration sets is related to the solution of a geometric problem of independent interest, namely, the problem of determining the existence of curvature-bounded paths traversing polygonal regions that may be partitioned into a sequence of rectangles. In Appendix C, we discuss this problem for rectangles of arbitrary dimensions. The proposed approach is less conservative than the best known result for curvature-bounded traversal across narrow channels [14], i.e., using the proposed approach, we can establish the existence of curvature-bounded paths in channels for which the result provided in [14] is inconclusive.

1.3.3 Multi-resolution path- and Motion Planning

As a specific example of implementation of the discrete path planner, we consider multi-resolution planar cell decompositions such that the environment is represented with high accuracy (i.e., with small cell sizes) in the agent's immediate vicinity, and with lower accuracy in regions farther away, similar to the multi-resolution grids considered in [11, 148]. This approach relates to two perspectives on the practical implementations of path planning: firstly, this approach relates to the problem of appropriately compressing *too much information* about the environment to enable efficient online computation. To this end, the proposed multi-resolution cell decomposition significantly reduces the number of vertices in the cell decomposition graph. On the other hand, this approach also relates to the problem of *uncertainty or partial knowledge* about the environment in regions farther away from the vehicle's location. To this end, the proposed path planning scheme requires accurate environment information only locally.

In light of the widespread use of the wavelet transform in signal processing and, by consequence, in autonomous navigation (as discussed in Section A.4), Tsiotras and Bakolas [147] and Jung [69] discuss a path planning scheme that directly uses a wavelet transform representation of the environment. This approach allows for the future development of highly efficient navigation and path planning schemes, where the wavelet transform coefficients may be used for signal analysis in navigation schemes while simultaneously serving as a data structure for path planning.

In this thesis, we propose a modification of the path planning scheme developed in [69, 147], and we rigorously prove the completeness of the proposed scheme. Such proofs of completeness are altogether absent from other works on multi-resolution path planning (cf. [11, 69, 147, 148]).

Furthermore, we incorporate vehicle dynamical constraints in the multi-resolution path planning scheme using the H-cost approach previously discussed. In particular,

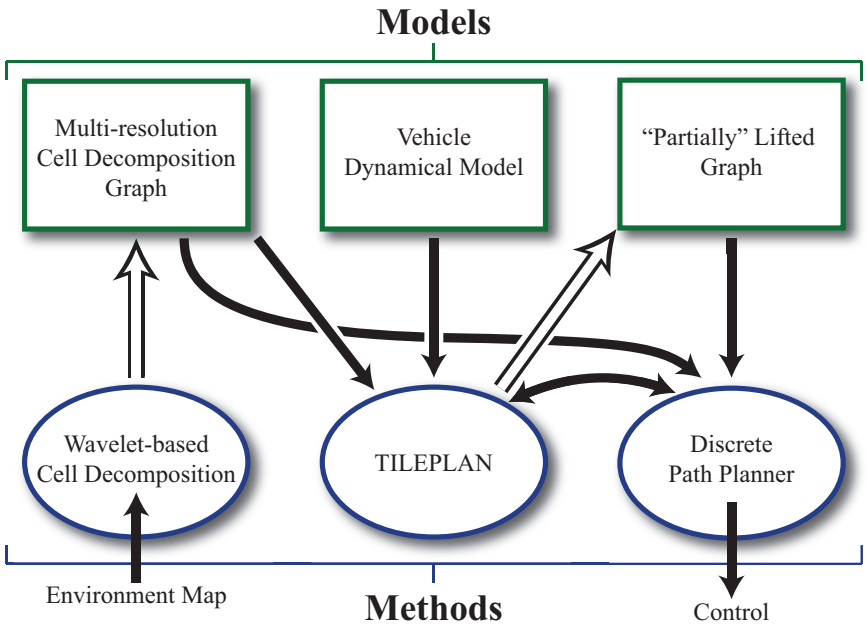


Figure 1.6: Schematic illustration of the overall motion planning framework. The hollow arrows indicate modification of a model by a method. The bold arrows indicate dependencies between the various methods and models.

we discuss a motion planner based on a "partially" lifted cell decomposition graph, which is "multi-resolution" both in the sense of representing the environment with high accuracy only locally, *and* in the sense of considering the vehicle dynamical constraints for path planning only locally.

1.3.4 Summary of Contributions

The motion planning framework proposed in this thesis is illustrated schematically in Fig. 1.6. In brief summary, the main contributions of this thesis are:

1. A *motion planning framework based on history-dependent costs* (H-costs) in cell decomposition graphs for incorporating vehicle dynamical constraints: this framework offers distinct advantages in comparison with the competing approaches of discretization of the state space, of randomized sampling-based motion planning, and of local feedback-based, decoupled hierarchical motion planning,

2. An efficient and flexible *algorithm for finding optimal H-cost paths*,

3. A precise and general *formulation of a local trajectory problem* (the tile motion planning problem) that allows independent development of the discrete planner and the trajectory planner, while maintaining "compatibility" between the two planners,

4. A local trajectory generation algorithm using MPC, and the *application of the concept of effective target sets* for a significant simplification of the local trajectory generation problem,

5. The *geometric analysis of curvature-bounded traversal* of rectangular channels, leading to less conservative results in comparison with a result reported in the literature [14], and also to the efficient construction of effective target sets for the solution of the tile motion planning problem,

6. A wavelet-based multi-resolution path planning scheme, and a *proof of completeness* of the proposed scheme: such proofs are altogether absent from other works on multi-resolution path planning,

7. A technique for extracting *all* information about cells – namely, the locations, the sizes, and the associated image intensities – directly from the set of significant detail coefficients considered for path planning at a given iteration, and

8. The extension of the multi-resolution path planning scheme to include vehicle dynamical constraints using the aforementioned H-cost approach.

Chapter 2

History-dependent Costs in Path Planning

Geometric path planning algorithms based on workspace cell decompositions provide no guarantees that the resultant channel of cells can be feasibly traversed by a vehicle subject to kinematic and dynamical constraints[1]. At first glance, one may argue that this is but an artefact of an inappropriate choice of the edge cost function in the associated graph. We provide a counter-example against this argument.

Consider the path planning problem depicted in Fig. 2.1, where S denotes the initial position, G denotes the goal, and the dark areas are obstacles. Consider two vehicles A and B, whose minimum radii of turn are kinematically constrained by $r^{\mathrm{A}}_{\mathrm{min}}$ and $r^{\mathrm{B}}_{\mathrm{min}}$ respectively, such that $r^{\mathrm{A}}_{\mathrm{min}} \leqslant d$ and $r^{\mathrm{B}}_{\mathrm{min}} > d$. Clearly, the dashed path in Fig. 2.1 is feasible for vehicle A, but not for vehicle B.

Figure 2.2 depicts the same problem with a uniform cell decomposition. The channel containing the dashed path of Fig. 2.1 is denoted by cells with bold outlines. Such a channel is obviously not traversable by vehicle B. However, notice that no *pair of successive cells* is *by itself* infeasible, i.e., a channel defined by two successive cells alone always contains a feasible path. Stated differently, for any two adjacent cells, there is no cell-dependent property associated with the two adjacent cells that can be penalized by an edge cost function in order to prevent the graph search from generating a channel such as the one shown in Fig. 2.2(a).

It may be further argued that a feasible path is guaranteed to exist in *any* channel if the dimensions of the cells are large enough. Indeed, Ref. [14] shows that a curvature-bounded path with local curvature less than or equal to $1/r_{\mathrm{min}}$ exists in a

[1] As previously mentioned, state space decompositions can avoid this problem but are impractical for high dimensional state spaces.

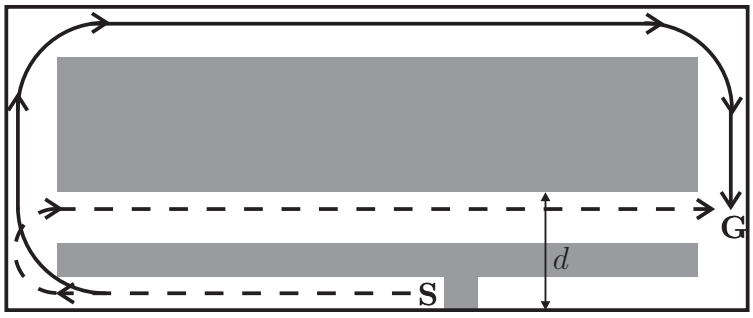
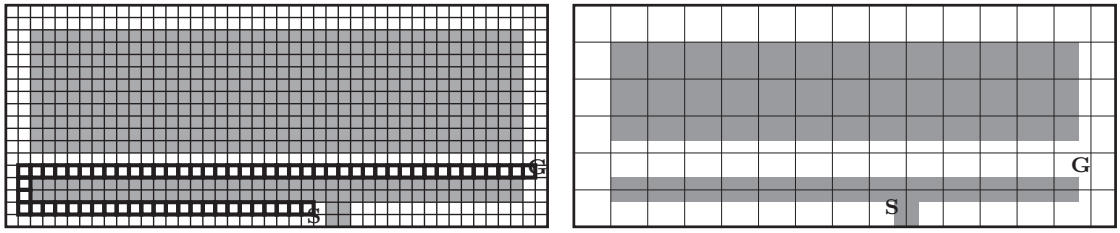


Figure 2.1: Counter-example for path planning without kinematic constraints.



(a) No pair of successive cells is itself infeasible. (b) Cells are too large, all cells are MIXED.

Figure 2.2: Problems with geometric path planning using cell decompositions.

polygonal channel if the width w of the channel satisfies $w \geqslant \tau r_{\min}$, where $\tau \approx 1.55$. The above counter-example also serves to illustrate that such a choice of cells may be too restrictive in practice. Figure 2.2(b) shows that large cells may not adequately capture the details of the environment, i.e., the number of MIXED cells could be too large for the cell decomposition to be useful for path planning. Note that this observation is consistent with the fact that nonholonomicconstraints (which lead to the curvature constraints) cannot be integrated to obtain equivalent position constraints.

In light of the preceding observations, we propose in this thesis an approach to find paths in the cell decomposition that minimize a cost defined on multiple edge transitions – called *histories* – instead of costs defined on single edge transitions.

2.1 Problem Formulation

In this section, we consider a precise formulation of the first (discrete) sub-problem of the hybrid point-to-point motion planning problem (Problem 1.2).

Consider a cell decomposition $\mathfrak{C}$ of the workspace; for now, we make no assumptions about the geometry of the cells involved in this decomposition. The topological graph associated with the cell decomposition $\mathfrak{C}$ is a graph $\mathcal{G} := (V, E)$, such that each element in the set of vertices V corresponds to a unique cell. Two vertices are *adjacent* if the corresponding cells are geometrically adjacent. The edge set E is the collection of all two-element subsets $\{i, j\} \subset V$ such that the vertices i and j are adjacent.

For given initial and goal vertices i_{S}, $i_{\mathrm{G}} \in V$, an *admissible path* π in the graph $\mathcal{G}$ is a finite sequence $(j_0, j_1, \ldots, j_P)$ of vertices (with no repetition) such that $j_k \in V$ and $\{j_{k-1}, j_k\} \in E$, for each $k = 1, \ldots, P$, with $j_0 = i_{\mathrm{S}}$ and $j_P = i_{\mathrm{G}}$. We introduce an edge cost function $g_0 : E \to \mathbb{R}_+$, which assigns to each edge of $\mathcal{G}$ a non-negative cost of transitioning this edge. The *standard shortest path problem* on the graph $\mathcal{G}$ is then defined as follows.

Problem 2.1 (Standard Shortest Path Problem)**.** *Let $i_{\mathrm{S}}, i_{\mathrm{G}} \in V$ be given initial and goal vertices. The standard cost of an admissible path $\pi = (j_0, \ldots, j_P)$ in the graph $\mathcal{G}$ is defined by*

$$\mathcal{J}_0(\pi) \;:=\; \sum_{k=1}^{P} g_0\left(j_{k-1}, j_k\right). \tag{2.1}$$

Find an admissible path π^* in the graph $\mathcal{G}$ such that $\mathcal{J}_0(\pi^*) \leqslant \mathcal{J}_0(\pi)$ for every admissible path π in the graph $\mathcal{G}$.

We introduce next a shortest path problem with costs defined on multiple edge transitions (histories). To formalize the concept of histories, we define, for every integer $H \geqslant 0$, the set

$$\begin{aligned} V_H \;:=\; & \{(j_0, \ldots, j_H) : \{j_{k-1}, j_k\} \in E,\ k = 1, \ldots, H, \\ & j_k \neq j_m, \ \text{ for } k, m \in \{0, \ldots, H\}, \ \text{with } k \neq m\}. \end{aligned}$$

An element of the set V_{H+1} is called a *H-history*. Let $I \in V_H$; in what follows, we will denote by $[I]_k$ the k^{th} element of this $(H+1)$-tuple, and by $[I]_k^\ell$ the tuple $([I]_k, [I]_{k+1}, \ldots, [I]_\ell)$, for $k < \ell \leqslant H+1$. We associate with each H a non-negative cost function $g_H : V_{H+1} \to \mathbb{R}_+$, and state a shortest path problem with transition costs defined on histories as follows.

Problem 2.2 (H-Cost Shortest Path Problem)**.** *Let $H \geqslant 0$, and let $i_{\mathrm{S}}, i_{\mathrm{G}} \in V$ be given initial and goal vertices, such that any admissible path in $\mathcal{G}$ contains at least $H+1$ vertices. The H-cost of an admissible path $\pi = (j_0, \ldots, j_P)$ in $\mathcal{G}$ is defined by*

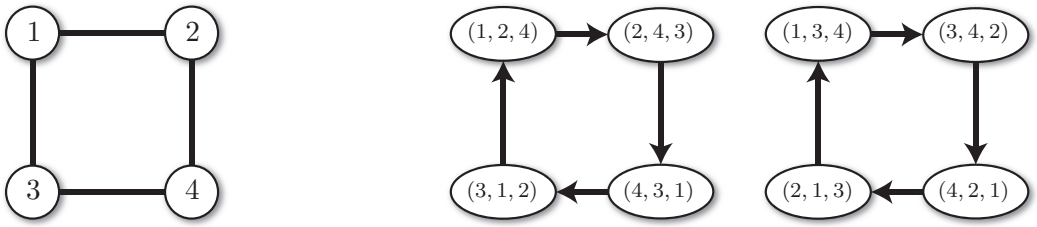
$$\mathcal{J}_H(\pi) \;:=\; \sum_{k=H+1}^{P} g_H\left((j_{k-H-1}, j_{k-H}, \ldots, j_k)\right). \tag{2.2}$$

Find an admissible path π^* in the graph $\mathcal{G}$ such that $\mathcal{J}_H(\pi^*) \leqslant \mathcal{J}_H(\pi)$ for every admissible path π in the graph $\mathcal{G}$.

Note that the H-cost of a path is defined as the sum of the costs of H-histories in that path. According to this convention, the 0-cost of a path is the standard cost defined in (2.1) because each 0-history in V_1 is associated with an unique edge in E. In other words, the H-cost shortest path problem for $H = 0$ is the standard shortest path problem (Problem 2.1).

It is possible to transform Problem 2.2 into an equivalent standard shortest path problem on a *lifted graph* $\mathcal{G}_H$. This transformation enables a clear conceptualization of our proposed algorithm to solve Problem 2.2 in light of the fact that the solutions to the standard shortest path problem are well-known (as we shall discuss in the next section). The vertices of the lifted graph $\mathcal{G}_H$ are the elements of V_H, and the edge set E_H of the lifted graph $\mathcal{G}_H$ is the set of all ordered pairs (I, J), such that $I, J \in V_H$, with $[I]_k = [J]_{k-1}$, for every $k = 2, \ldots, H+1$, and $[I]_1 \neq [J]_{H+1}$. The notion of the lifted graph is illustrated in Fig. 2.3 for $H = 1$.

For given initial and terminal vertices i_{S}, $i_{\mathrm{G}} \in V$, an admissible path Π in $\mathcal{G}_H$ is a finite sequence $(J_0, \ldots, J_Q)$ of vertices (with no repetition) such that $(J_{k-1}, J_k) \in$



(a) Original graph $\mathcal{G}$ (b) Lifted graph $\mathcal{G}_1$

Figure 2.3: Illustrative example of the lifted graph with $H = 1$.

E_H, for each $k = 1, \ldots, Q$, with $[J_0]_1 = i_{\mathrm{S}}$, and $[J_Q]_{H+1} = i_{\mathrm{G}}$. Note that every admissible path $\Pi = (J_0, \ldots, J_Q)$ in $\mathcal{G}_H$ uniquely corresponds to an admissible path $\pi = (j_0, \ldots, j_P)$ in $\mathcal{G}$, with $P = Q+H$ and $[J_k]_m = j_{kH+m-1}$, for each $k = 0, 1, \ldots, Q-1$, and $J_Q = (j_{P-H}, \ldots, j_P)$. We introduce a non-negative cost function $\tilde{g}_H : E_H \to \mathbb{R}_+$ defined by $\tilde{g}_H((I, J)) := g_H\left(([I]_1^{H+1}, [J]_{H+1})\right)$, for every pair $(I, J) \in E_H$. It follows that Problem 2.2 is equivalent to the standard shortest path problem on the graph $\mathcal{G}_H$, where the cost of an edge $(I, J) \in E_H$ given by $\tilde{g}_H((I, J))$.

The lifted graph $\mathcal{G}_H$ is a discretization of the workspace with richer properties in the context of motion planning, in that it can represent the vehicle kinematic/dynamic constraints, as we shall clarify in what follows.

2.2 Path Planning with History-Dependent Costs

The standard shortest path problem can be solved by a class of algorithms called the *label-correcting algorithms*. Well-known examples of label correcting algorithms include the Bellman-Ford, the Dijkstra [15, 34], and the A* [93, 112] algorithms.

A label-correcting algorithm progressively searches for the least cost path starting from i_{S} and ending at vertex $i \in V$, by iteratively reducing an estimate of the least cost to the initial vertex i, called the *label* of the vertex i. Let $d : V \to \mathbb{R}_+$ denote the label function, i.e., $d(i)$ is the current estimate of the least cost from i_{S} to i. The algorithm also maintains a set $\mathcal{P}$ of vertices, called the *fringe* [130], that contains the vertices whose labels can potentially be reduced from their current value (referred to

General Label-Correcting Algorithm

Input: $i_{\mathrm{S}} \in V$, **Output:** Label d, Backpointer b

procedure INITIALIZE(i_{S})

1: $\mathcal{P} \leftarrow \{i_{\mathrm{S}}\}, \quad d(i_{\mathrm{S}}) \leftarrow 0$
2: **for all** $j \in V \backslash \{i_{\mathrm{S}}\}$ **do**
3: $\quad d(j) = \infty$

procedure MAIN

1: INITIALIZE(i_{S})
2: **while** $\mathcal{P} \neq \varnothing$ **do**
3: $\quad i \leftarrow$ REMOVE($\mathcal{P}$)
4: $\quad$ **for all** $j \in V$ such that $\{i, j\} \in E$ **do**
5: $\quad\quad$ **if** $d(i) + g(i, j) < d(j)$ **then**
6: $\quad\quad\quad d(j) \leftarrow d(i) + g(i, j)$
7: $\quad\quad\quad b(j) \leftarrow i$
8: $\quad\quad\quad \mathcal{P} \leftarrow$ INSERT($\mathcal{P}, j$)

Figure 2.4: Pseudo-code for the general label-correcting algorithm.

as the OPEN list in [15,34]), and it maintains a *backpointer* function $b : V \to V$, which records the immediate predecessor of each node $i \in V$ in the optimal path from i_{S} to i. The pseudo-code for a general label-correcting algorithm is shown in Fig. 2.4.

The H-cost shortest path problem can be solved by first transforming it to a standard shortest path problem on the graph $\mathcal{G}_H$, and then executing a standard label-correcting algorithm such as Dijkstra's algorithm. A naïve, brute-force implementation of this approach is ill-advised because (a) $|V_H|$ and $|E_H|$ grow exponentially with H, and (b) the explicit construction of the graph $\mathcal{G}_H$ may be unnecessary to find the shortest path in $\mathcal{G}_H$.

In this section, we describe an algorithm that is equivalent to executing a label-correcting algorithm on the graph $\mathcal{G}_H$; however, the proposed approach does not construct the entire graph $\mathcal{G}_H$ beforehand. Since $|V_H|$ and $|E_H|$ grow exponentially

with H, the execution time of *any* algorithm that solves the H-cost shortest path problem *exactly* grows exponentially with H. This fact holds true for the proposed algorithm; however, we include in our algorithm a user-specified parameter that can dramatically reduce the execution time at the expense of (exact) optimality of the resultant path. In other words, the proposed algorithm exhibits a flexibility that allows the user to trade off execution time against optimality of the resultant path.

For the sake of clarity, we present first a basic version of our algorithm, namely, one that finds the optimal path and solves the H-cost shortest path problem exactly. In Section 2.3.1, we introduce the aforementioned user-specified parameter and discuss the effects of this parameter on the algorithm's execution time.

Recall that the standard label-correcting algorithm maintains a collection of vertices, namely, the fringe, whose labels can potentially be reduced, and that it associates with each vertex a label and a backpointer. Also, observe that the definition of an admissible path in $\mathcal{G}_H$ from i_{S} to any vertex $i \in V$ requires only $[J_Q]_{H+1} = i$, where $J_Q \in V_H$ is the last vertex in this path. The first H elements of J_Q are unspecified, which implies that different admissible paths in $\mathcal{G}_H$ may have different terminal vertices in V_H. In the proposed algorithm, we recognize this fact by associating with each vertex $i \in V$ *multiple* H-histories instead of the (single) backpointer in the standard label-correcting algorithm. Each history of i is a unique element $I \in V_{H+1}$ such that $[I]_{H+2} = i$. The proposed algorithm is a label-correcting algorithm that associates with each history of each vertex $i \in V$ a label. Accordingly, the fringe in the proposed algorithm is a collection of *pairs*, where each pair consists of a vertex in V and an index that refers to a particular history of that vertex.

A detailed pseudo-code of the proposed algorithm is presented in Fig. 2.5. At each iteration, the algorithm updates the label corresponding to a vertex-index pair, i.e., a particular history. Lines 8–11 update the fringe, the labels, and the histories, similar to Lines 4-8 of the standard label-correcting algorithm in Fig. 2.4. Line 5 chooses the

H-Cost Shortest Path Algorithm

Input: $i_{\mathrm{S}} \in V$, **Output:** Label d, History h

procedure INITIALIZE(i_{S})

for all $i \in V$, $m = 1, \ldots, |\mathcal{H}_i|$ **do**
  $d(i,m) \leftarrow \infty, \quad h(i,m) \leftarrow$ NULL
  $\mathcal{P} \leftarrow \{(i,m) : \mathcal{N}_i \neq \varnothing, \text{ and } \textsc{History}(i,m) \in \mathcal{N}_i\}$
for all $(i,m) \in \mathcal{P}$ **do**
  $I \leftarrow \textsc{History}(i,m)$
  $d(i,m) \leftarrow \tilde{g}_{H+1}(I)$, $h(i,m) \leftarrow [I]_1^{H+1}$

procedure MAIN

INITIALIZE(i_{S})
while $\mathcal{P} \neq \varnothing$ **do**
  $(i,m) \leftarrow \textsc{Remove}(\mathcal{P})$
  for all $j \in V$ such that $(i,j) \in E$ **do**
    $n \leftarrow \textsc{Index}(([h(i,m)]_3^{H+1}, i, j), j)$
    $J \leftarrow \textsc{History}(j,n)$
    $D_{i,m} \leftarrow d(i,m) + \tilde{g}_{H+1}([h(i,m)]_2, J)$
    if $d(j,n) > D_{i,m}$ **then**
      $d(j,n) \leftarrow D_{i,m}$
      $h(j,n) \leftarrow ([h(i,m)]_2^{H+1}, i)$
      $\mathcal{P} \leftarrow \textsc{Insert}(\mathcal{P}, (j,n))$

Figure 2.5: Pseudo-code for the basic version of the proposed algorithm.

index corresponding to the particular history (of the newly explored vertex j) being updated in that iteration. We use the following notation: for each vertex $i \in V$, $\mathcal{H}_i$ and $\mathcal{N}_i$ are defined by

$$\mathcal{H}_i := \{I \in V_H : [I]_{H+1} = i\}, \tag{2.3}$$

$$\mathcal{N}_i := \{I \in V_{H+1} : [I]_1 = i_{\mathrm{S}}, [I]_2^{H+2} \in \mathcal{H}_i\}. \tag{2.4}$$

Here $\mathcal{P}$ denotes the fringe; for each vertex $i \in V$ and $m \in \{1, \ldots, \mathcal{H}_i\}$, $h(i,m)$ denotes the m^{th} history of i and $d(i,m)$ denotes the label associated with the m^{th} history of i.

Finally, the procedure HISTORY(i, m) returns the m^{th} element of the set $\mathcal{H}_i$; and the procedure INDEX(I, i) returns the index of the history I in the set $\mathcal{H}_i$ if $I \in \mathcal{H}_i$, or returns 0 otherwise. The procedures INSERT and REMOVE depend on the specific data structure used for implementing the fringe.

The algorithm terminates when $\mathcal{P} = \varnothing$ (in Section 2.3, we note a condition for terminating the algorithm earlier). After termination, the algorithm returns the labels d and the histories h. For every vertex $i \neq i_{\text{S}}$ in V, we may then calculate the optimal path from i_{S} to i by recursively tracing the $(H+1)^{\text{th}}$ element of histories recorded by h: a process similar to recursively tracing the backpointer in the standard label-correcting algorithm. We illustrate the execution of the algorithm with a simple example.

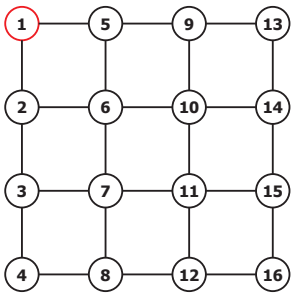
Example 2.3. Consider the graph shown in Fig. 2.6(a), where $i_{\text{S}} = 1$. Let $H = 1$. Let $\tilde{g}_2$ be a non-negative cost function given by the lookup table in Fig. 2.6(b) (for brevity, the values of some of the elements of V_2 are not shown). Note that

$$|\mathcal{H}_i| \;=\; \begin{cases} 2, & i \in \{1, 4, 13, 16\}, \\ 3, & i \in \{2, 3, 5, 8, 9, 12, 14, 15\}, \\ 4, & \text{otherwise}, \end{cases}$$

i.e., the algorithm maintains at most 4 histories and labels for each vertex. Also note that $\mathcal{N}_3 = \{(1, 2, 3)\}$, $\mathcal{N}_6 = \{(1, 5, 6), (1, 2, 6)\}$, $\mathcal{N}_9 = \{(1, 5, 9)\}$, and $\mathcal{N}_i = \varnothing$ for $i \in \{1, \ldots 16\} \backslash \{3, 6, 9\}$.

We index the elements of $\mathcal{H}_j$ as $1, 2, 3, 4$ corresponding to UP, RIGHT, DOWN, LEFT edges of j, with reference to Fig. 2.6(a). If a particular edge is absent, the corresponding index applies to the next edge in the order listed above. For example, the indices of $(5, 6), (10, 6), (7, 6), (2, 6) \in \mathcal{H}_6$ are 1, 2, 3, and 4 respectively, while the indices of $(5, 1), (2, 1) \in \mathcal{H}_1$ are 1 and 2 respectively.

Line 3 of procedure INITIALIZE results in $\mathcal{P} = \{(3, 1), (6, 1), (6, 4), (9, 3)\}$. By Fig. 2.6(b), Line 6 of procedure INITIALIZE results in



(a) Graph $\mathcal{G}$

$I \in V_2$	$g_1(I)$	$I \in V_2$	$g_1(I)$
$(1,2,3)$	5	$(5,6,2)$	5
$(1,2,6)$	6	$(5,6,7)$	7
$(1,5,6)$	2	$(5,6,10)$	8
$(1,5,9)$	8	$(2,6,5)$	12
$(2,6,10)$	5	$(2,6,7)$	8

(b) H-costs for $H = 1$

Figure 2.6: The graph and the history-based cost function used in Example 1.

$$d(3,1) = 5, \quad d(6,1) = 2, \quad d(6,4) = 6, \quad d(9,3) = 8,$$
$$h(3,1) = (1,2), \quad h(6,1) = (1,5), \quad h(6,4) = (1,2), \quad h(9,3) = (1,5).$$

Next, suppose we remove $(i,m) = (6,1)$ from the fringe in Line 3 of procedure MAIN. Then $\mathcal{P} = \{(3,1),(6,4),(9,3)\}$, and the **for** loop in Line 4 is executed for vertices 2, 5, 7 and 10. In particular, for vertex $j = 2$, Lines 5 and 6 result in $n = 2$, and $J = (6,2)$, respectively. By Table 2.6(b), it follows that $d(6,1)+g_1([h(6,1)]_2,6,2) = 2+5 = 7 < d(2,2) = \infty$ (by Line 2 of procedure INITIALIZE). Hence, Lines 9 and 10 result in $d(2,2) = 7$, and $h(2,2) = ([h(6,1)]_2,6) = (5,6)$, while Line 11 results in $\mathcal{P} = \{(3,1),(6,4),(9,3),(2,2)\}$.

Similarly, the execution of the **for** loop in Line 4 of procedure MAIN for vertices $5, 7$, and 10, results in $\mathcal{P} = \{(3,1),(6,4),(9,3),(2,2),(7,1),(10,4),(5,2)\}$. The labels and histories at the end of the first iteration are

$$d(5,2) = 18, \quad d(7,1) = 9, \quad d(10,4) = 10,$$
$$h(5,2) = (2,6), \quad h(7,1) = (5,6), \quad h(10,4) = (5,6).$$

□

2.3 Theoretical Analysis and Numerical Simulation Results

Different instances of label-correcting algorithms are obtained by implementing the fringe using different data structures. For example, implementing the fringe as a LIFO stack results in a breadth-first search; implementing the fringe as a list sorted by current labels results in Dijkstra's algorithm. In this section, we consider an instance

of the proposed algorithm with the fringe implemented as a list sorted by the current labels, i.e., the procedure REMOVE returns $(i, m) = \arg\min\{d(i, m) : (i, m) \in \mathcal{P}\}$.

Proposition 2.4. *For every vertex $i \in V$, suppose there exists at least one admissible path from i_{S} to i containing* HISTORY(i, m)*, for any $m \in \{1, \ldots, |\mathcal{H}_i|\}$. Let π^* be such an admissible path in $\mathcal{G}$ with the least cost. Then the proposed algorithm terminates with $d(i, m) = \tilde{\mathcal{J}}_H(\pi^*)$. Otherwise, the algorithm terminates with $d(i, m) = \infty$.*

Proof. See Appendix B. □

Proposition 2.4 asserts that the algorithm computes the minimum H-cost of paths from i_{S} to every vertex $i \in V$ and every history in $\mathcal{H}_i$. However, because we need to compute only the minimum H-cost from i_{S} to i_{G} for any history in $\mathcal{H}_{i_{\mathrm{G}}}$, it is possible to terminate the algorithm earlier, as shown by the following result.

Proposition 2.5. *Each pair (j, m), $j \in V$, $m = 1, \ldots, |\mathcal{H}_i|$ enters the set $\mathcal{P}$ at most once during the execution of the algorithm.*

Proof. See Appendix B. □

The conditions in Lines 8 and 11 of the procedure MAIN imply that a pair (j, m) is inserted in $\mathcal{P}$ only when the value of $d(j, m)$ can be reduced. It follows from Proposition 2.5 that once a pair (j, m) is removed from $\mathcal{P}$, the value of $d(j, m)$ cannot be further reduced. The implication of this fact is that we may terminate the algorithm after Line 3 if $i = i_{\mathrm{G}}$.

Note that Proposition 2.5 closely resembles a similar, known result regarding the execution of Dijkstra's algorithm: namely, that each vertex enters the fringe at most once (see, for instance, [15, 34]). We may use Proposition 2.5 to characterize the execution time of the basic version of the proposed algorithm as follows: in the worst case, every pair (j, m) enters the set $\mathcal{P}$ exactly once. Thus, the maximum

number of iterations (of the **while** loop in procedure MAIN) is upper bounded by $\sum_{j=1}^{|V|} |\mathcal{H}_j| = |V_H| = O(|V|^H)$.

2.3.1 Modifications for Further Efficiency

As mentioned previously, the execution time of the basic version of the proposed algorithm increases exponentially with H, which may slow down the algorithm for large values of H. To address this issue, we present in this section a simple modification of the basic version of the algorithm that dramatically reduces its execution time at the expense of optimality of the resultant path.

The algorithm presented in Fig. 2.5 maintains, for each node $j \in V$, a record of the costs-to-come to that node through each of its histories. To reduce the execution time of the algorithm, we may modify the algorithm such that it maintains, for each node $j \in V$, the costs-to-come through a fixed number L of histories. In particular, we modify the proposed algorithm by inserting the following statements between Lines 5 and 8 in procedure MAIN:

$\mathcal{L}_i \leftarrow \{d(i,m) < \infty : m = 1, \ldots, |\mathcal{T}_i|\}$

if $|\mathcal{L}_i| = L$ and $D_{i,m} \geqslant \max\{\mathcal{L}_i\}$ **then**

  continue

Accordingly, we delete from the set $\mathcal{N}_i$ defined in (2.4) all but the first L histories, ranked by increasing H-costs.

2.3.2 Numerical Simulation Results

Figure 2.7 shows, on a logarithmic scale, the maximum, the minimum, and the average ratios of the time required for constructing the lifted graph $\mathcal{G}_H$ and then executing Dijkstra's algorithm to the execution time of the execution time of the proposed algorithm. Each data point in Fig. 2.7 corresponds to a different combination of H and the size $|\mathcal{G}|$ of the cell decomposition graph. The specific values for H and $|\mathcal{G}|$

corresponding to the data points from left to right in Fig. 2.7 are as follows:

$$
\begin{aligned}
H = 1, &\quad |\mathcal{G}| = N^2, \text{ for each } N = 10, 20, \ldots, 150, \\
H = 2, &\quad |\mathcal{G}| = N^2, \text{ for each } N = 10, 20, \ldots, 130, \\
H = 3, &\quad |\mathcal{G}| = N^2, \text{ for each } N = 10, 20, \ldots, 100, \\
H = 4, &\quad |\mathcal{G}| = N^2, \text{ for each } N = 10, 20, \ldots, 60, \\
H = 5, &\quad |\mathcal{G}| = N^2, \text{ for each } N = 5, 15, 25, 35, \\
H = 6, &\quad |\mathcal{G}| = N^2, \text{ for each } N = 10, 15, 20, 25, \\
H = 7, &\quad |\mathcal{G}| = N^2, \text{ for each } N = 5, 10, 15.
\end{aligned}
$$

Table 2.1 shows sample values of the data represented in Fig. 2.7: the fourth, fifth, and sixth columns show, respectively, the absolute values of the maximum, the minimum, and average ratios of execution times. The graph $\mathcal{G}$ used in these simulations is the graph arising out of a uniform cell decomposition with 4-connectivity, i.e., a graph of the form shown in Fig. 2.6(a). For each combination of H and $|\mathcal{G}|$, 30 trials were performed. In each of these trials, the structure of the graph $\mathcal{G}$ was kept constant and the costs of transitioning H-histories, i.e., the costs of edge transitions in $\mathcal{G}_H$, were assigned randomly. The initial and goal nodes i_{S} and i_{G} were randomly assigned in each trial.

The simulation results shown in Fig. 2.7 and Table 2.1 indicate that the proposed algorithm executes up to three orders of magnitude faster on average, and may execute up to four orders of magnitude faster in the best-case, than the alternative approach of first constructing the lifted graph and then executing the search. Furthermore, the memory required to store the graph $\mathcal{G}_H$ is approximately K times the memory required by the proposed algorithm to store multiple histories of each node $j \in V$, where K is the valency of the graph $\mathcal{G}$. Note, however, that the *minimum* ratios of execution times are close to unity in a large number of the cases, indicating that the *complexity* of the H-cost shortest path algorithm is the same as that of the alternative approach of executing Dijkstra's algorithm on the lifted graph.

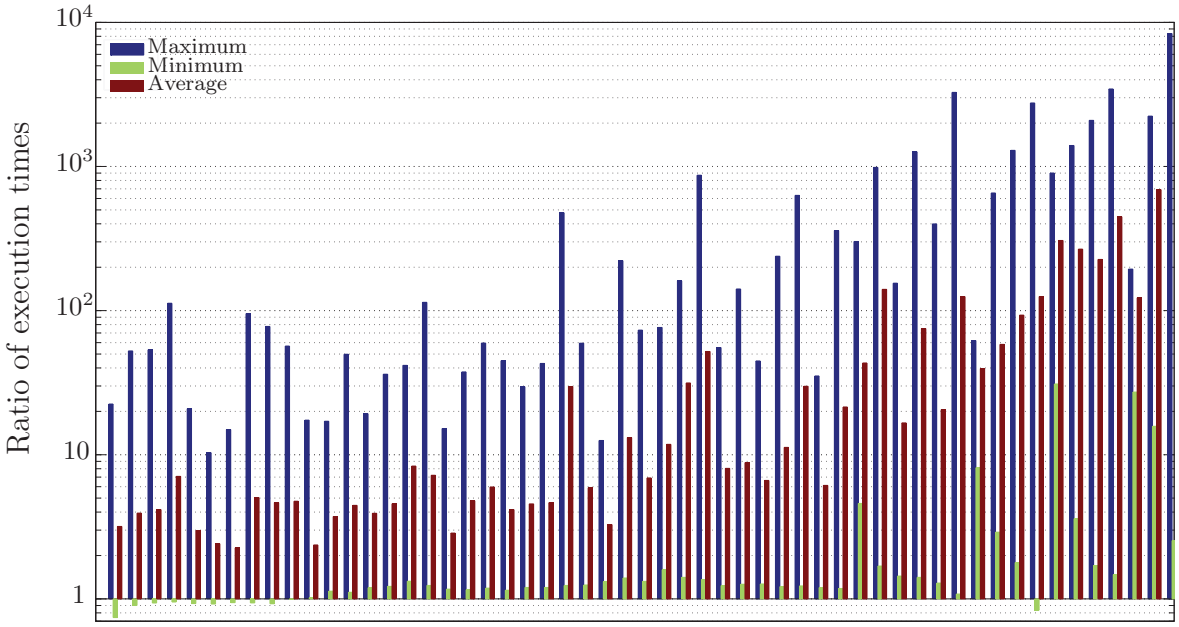


Figure 2.7: Comparison of execution time of exact H-cost shortest path algorithm with the execution of Dijkstra's algorithm on the lifted graph.

$\lvert\mathcal{G}\rvert$	H	$\lvert\mathcal{G}_H\rvert$	Max. ratio	Min. ratio	Avg. ratio
6,400	1	25,280	95.49	0.937	5.039
10,000	1	39,600	56.78	0.993	4.756
14,400	1	57,120	17.07	1.017	3.726
22,500	1	89,400	36.19	1.221	4.574
6,400	2	74,888	43.01	1.199	4.646
10,000	2	117,608	59.43	1.249	5.901
14,400	2	169,928	222.4	1.396	13.15
16,900	2	199,688	73.27	1.322	6.901
2,500	3	85,056	141.3	1.263	8.831
4,900	3	169,456	238.4	1.215	11.26
6,400	3	222,456	630.7	1.231	29.87
10,000	3	350,056	359.7	1.182	21.47
900	4	79,472	155.1	1.440	16.63
1,600	4	145,872	1264	1.410	75.14
2,500	4	232,272	399.8	1.287	20.57
625	5	147,952	1294	1.788	93.06
1,225	5	306,072	2 761	0.834	125.2
225	6	120,532	1399	3.604	267.1
400	6	237,232	2091	1.697	226.6

Table 2.1: Comparison of execution time of exact H-cost shortest path algorithm with the execution of Dijkstra's algorithm on the lifted graph: sample values.

Figure 2.8 shows, on a logarithmic scale, the maximum, the minimum, and the average ratios of the execution time of the exact H-cost shortest path algorithm to the execution time of the modified H-cost shortest path algorithm of Section 2.3.1 and the corresponding sub-optimality of the resultant paths in terms of the percentage increase in cost. Each data point in Fig. 2.8 corresponds to a different combination of H, the size $|\mathcal{G}|$ of the cell decomposition graph, and the parameter L. The specific values for H, $|\mathcal{G}|$, and L corresponding to the data points from left to right in Fig. 2.8 are as follows:

$$
\begin{aligned}
&H = 1, \quad |\mathcal{G}| = 10,000, \quad 14,400, \quad 22,500, \quad && L = 1, 2, \\
&H = 2, \quad |\mathcal{G}| = 8,100, \quad 10,000, \quad 14,400, \quad && L = 1, 3, 5, \\
&H = 3, \quad |\mathcal{G}| = 4,900, \quad 8,100, \quad 10,000, \quad && L = 2, 4, 5, \\
&H = 4, \quad |\mathcal{G}| = 1,600, \quad 2,500, && L = 3, 5, 7, \\
&H = 5, \quad |\mathcal{G}| = 625, \quad 1,225 && L = 3, 5, 7, \\
&H = 6, \quad |\mathcal{G}| = 225, \quad 625 && L = 2, 5, 11.
\end{aligned}
$$

Table 2.2 shows a few sample cases of the data presented in Fig. 2.8. For each combination of H, $|\mathcal{G}|$, and L, 30 trials were performed. In each of these trials, the structure of the graph $\mathcal{G}$ was kept constant and the costs of transitioning H-histories, i.e., the costs of edge transitions in $\mathcal{G}_H$, were assigned randomly. The initial and goal nodes were kept fixed in each trial: in particular, $i_{\mathrm{S}} = 1$ and $i_{\mathrm{G}} = |\mathcal{G}|$ were assigned.

The simulation results shown in Fig. 2.8 and Table 2.2 indicate that relatively small values of L speed up the original algorithm by up to three orders of magnitude, with relatively low increases in the cost of resultant paths. A similar observation has been reported in [125], for the specific case of $H = 1$ and $L = 1$.

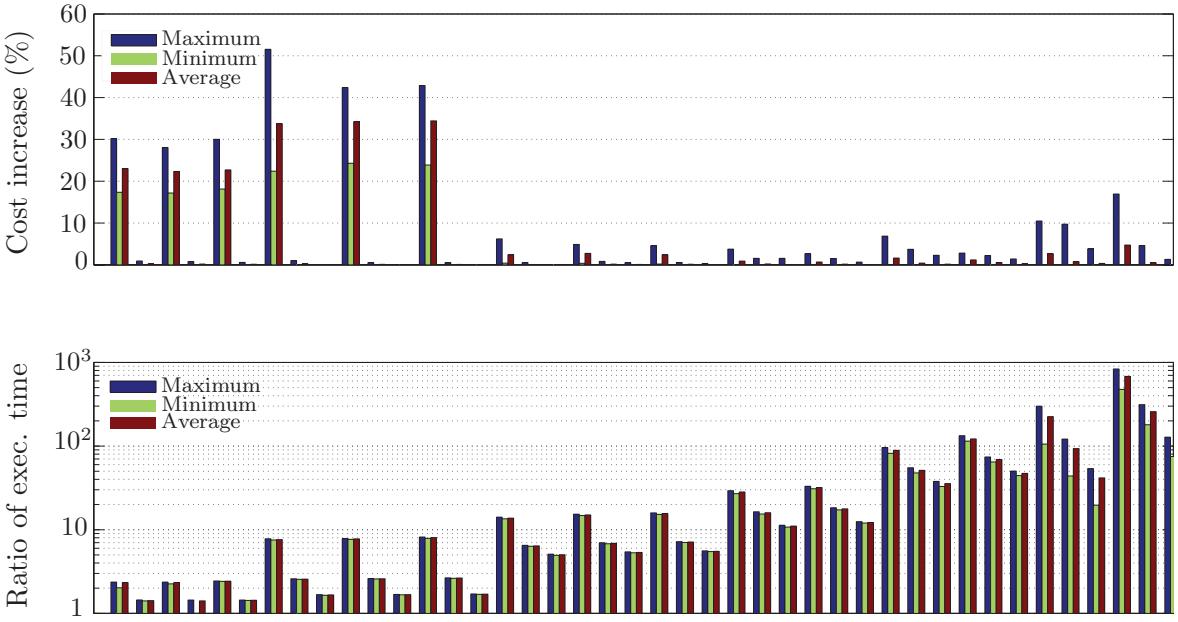


Figure 2.8: Comparisons of execution time and sub-optimality of the modified H-cost shortest path algorithm with the exact algorithm.

$\lvert\mathcal{G}\rvert$	H	L	$\max_i\{\lvert\mathcal{H}_i\rvert\}$	Time ratio	Cost diff. (%)
22,500	1	1	4	2.424	18.13
22,500	1	2	4	1.434	0.144
14,400	2	1	12	8.055	34.43
14,400	2	3	12	2.645	0.108
14,400	2	5	12	1.698	0.000
10,000	3	2	36	15.62	2.379
10,000	3	4	36	7.119	0.146
10,000	3	5	36	5.521	0.036
2,500	4	3	100	31.90	0.662
2,500	4	5	100	17.75	0.155
2,500	4	7	100	12.20	0.038
1,225	5	3	284	121.7	1.170
1,225	5	5	284	69.11	0.543
1,225	5	7	284	47.15	0.243
625	6	2	780	681.0	4.727
625	6	5	780	257.6	0.496
625	6	11	780	107.1	0.148

Table 2.2: Comparisons of execution time and sub-optimality of the modified H-cost shortest path algorithm with the exact algorithm: sample values.

Chapter 3

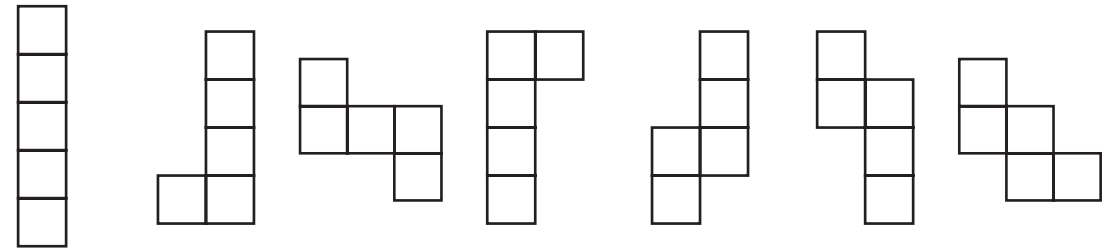
Motion Planning Framework based on H-Cost Shortest Paths

We develop in this chapter a hierarchical motion planning framework based on H-cost shortest paths. In this framework, the high-level geometric path planner repeatedly invokes a special trajectory generation algorithm, called the *tile motion planner*, to determine the costs of H-histories. To motivate the discussion and the potential benefits of the proposed motion planning framework, we first discuss a preliminary result of using H-costs in motion planning.

3.1 A Preliminary Result

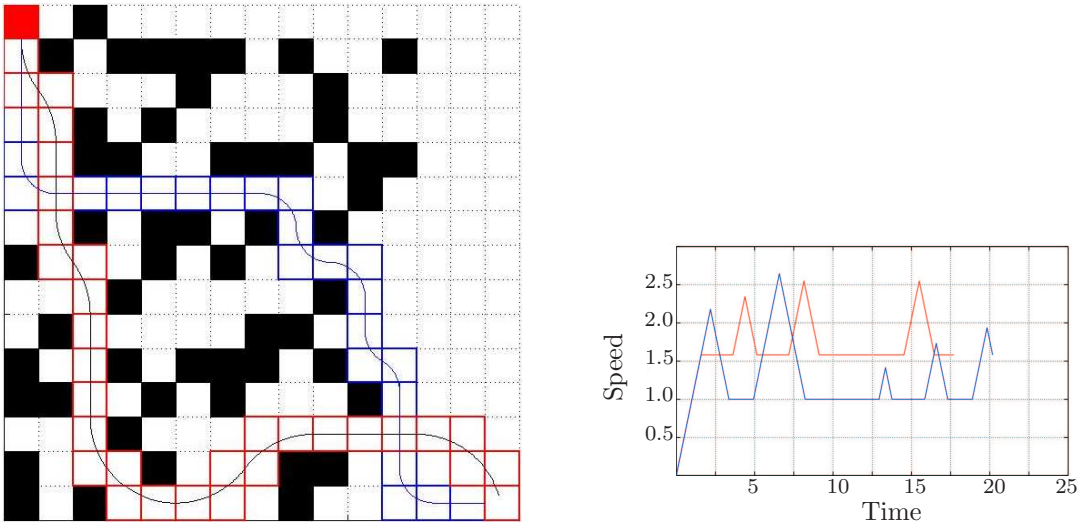
Consider two vertices $I, J \in V_H$ such that $(I, J) \in E_H$. We define the *tile* associated with the edge (I, J) as the sequence of cells associated with the tuple $([I]_1, [J]_1^{H+1})$ of vertices in $\mathcal{G}$. In this thesis, we will use the symbol $\mathcal{R}$ to denote a tile, and we will denote by $(I^{\mathcal{R}}, J^{\mathcal{R}})$ the edge in E_H associated with this tile. In this chapter, we consider an approximate cell decomposition consisting of uniformly sized (and spaced) squares.

For every $H \geqslant 1$, it is possible to identify a finite collection $\mathfrak{T}_H = \left\{\mathcal{R}_1^H, \mathcal{R}_2^H, \dots, \right\}$ of sequences of cells such that the tile associated with every edge in E_H is equivalent, up to rigid geometric transformations, to a unique element of the set $\mathfrak{T}_H$. Let $\phi : E_H \to \mathfrak{T}_H$ denote the map associating with each edge in E_H an element of $\mathfrak{T}_H$, and consider now a H-cost function defined as follows. Let $\mathfrak{T}_H^{\text{feas}} \subseteq \mathfrak{T}_H$ be a collection of sequences of cells which are deemed "feasible" following, perhaps, some *a priori*



(a) $\mathcal{R}_1^3$ (b) $\mathcal{R}_2^3$ (c) $\mathcal{R}_3^3$ (d) $\mathcal{R}_4^3$ (e) $\mathcal{R}_5^3$ (f) $\mathcal{R}_6^3$ (g) $\mathcal{R}_7^3$

Figure 3.1: Illustration of the sequences of cells in the collection $\mathfrak{T}_3^{\text{feas}}$.



(a) Environment (b) Optimal velocity profile

Figure 3.2: Preliminary result illustrating the potential benefits of motion planning based on H-cost path planning.

geometric considerations, and let g_3 be a H-cost function defined by

$$g_3((I,J)) := \begin{cases} 1, & \phi((I,J)) \in \mathfrak{T}^{\text{feas}}, \\ \infty, & \text{otherwise.} \end{cases} \tag{3.1}$$

We may define, for the sake of a concrete example, $\mathfrak{T}_3^{\text{feas}}$ as the collection of sequences of cells illustrated in Fig. 3.1.

In Fig. 3.2(a), the sequence of red-colored cells indicates the result of H-cost path planning with the H-costs defined by (3.1), whereas the blue-colored cells indicate the result of solving the standard shortest path problem on the cell decomposition graph with all edge costs equal to unity. The curvature of the red-colored curve is bounded above by $1/3$ units (the size of each cell is 1 unit), and this curve was fitted a

posteriori within the red-colored sequence of cells. On the other hand, the curvature of the blue-colored curve is bounded above by 1 unit, and it was not possible to fit a curve with maximum curvature 1/3 units within the blue-colored sequence of cells.

Most interestingly, Fig. 3.2(b) shows the corresponding optimal velocity profile from the solution of the minimum-time problem along each path [151]. In other words, although the red-colored path is longer, a vehicle traversing this path will require lesser time than a vehicle following the shorter (but with more and sharper turns) blue path. Furthermore, if the vehicle is kinematically constrained to traversing paths of curvature at most 1/3, then the blue-colored sequence of cells is *infeasible*.

3.2 General Hierarchical Motion Planning Framework

The H-cost function defined by (3.1) characterized *ad hoc* the feasibility of traversal of tiles. In this section, we discuss a general technique for assigning an H-cost to each edge in E_H based on the feasibility of traversal of the tile associated with that edge, considering the vehicle kinematic, dynamic, and control input constraints.

In what follows, we continue using the notation and terminology introduced in Section 1.1. Additionally, we will denote by $\mathsf{x}(\xi)$ the projection of the vehicle state $\xi \in \mathcal{D}$ on $\mathbb{R}^2$, i.e., the position components of the state ξ. Finally, we will denote by $\mathsf{cell}(i)$ the cell in $\mathfrak{C}$ associated with the vertex $i \in V$.

We define a special state trajectory planner called the *tile motion planner* (TilePlan) as a trajectory generation algorithm that determines if a given tile may be feasibly traversed by the vehicle from a specific initial state. The cost of traversal of a tile is the integral along the state trajectory of a pre-specified incremental cost $\ell(\xi, u, t)$. A precise and general definition of TilePlan is given in Fig. 3.3.

Briefly, TilePlan determines if there exists a finite time t_f and a control $u \in \mathcal{U}_{t_\mathrm{f}}$ such that the corresponding vehicle state trajectory satisfies the constraints (3.2) and (3.3). The constraint (3.2) states the requirement that the position components of

Tile Motion Planning Algorithm

Input: Tile $\mathcal{R}$, State ξ_0, **Output:** Time t_1, State ξ_1, Control $u_{[0,t_1]}$, Cost Λ

procedure TILEPLAN($\mathcal{R}, \xi_0$)

1: Determine if there exist $t_\mathrm{f} \in \mathbb{R}$ and admissible control input $u \in \mathcal{U}_{t_\mathrm{f}}$ such that $\xi(\cdot\,; \xi_0, u)$ satisfies

$$\mathsf{x}(\xi(t; \xi_0, u)) \in \textstyle\bigcup_{k=1}^{H} \mathsf{cell}([J^{\mathcal{R}}]_{k0}), \ t \in (0, t_\mathrm{f}), \tag{3.2}$$

$$\mathsf{x}(\xi(t_\mathrm{f}; \xi_0, u)) \in \mathsf{cell}([J^{\mathcal{R}}]_{H}) \cap \mathsf{cell}([J^{\mathcal{R}}]_{H+1}) \tag{3.3}$$

2: **if** $\exists t_\mathrm{f}$ and $\exists u$ **then**

3: Find t_1 such that

$$\mathsf{x}(\xi(t_1; \xi_0, u)) \in \mathsf{cell}([J^{\mathcal{R}}]_{1}) \cap \mathsf{cell}([J^{\mathcal{R}}]_{2}) \tag{3.4}$$

4: Return t_1, $u_{[0,t_1]}$, $\xi_1 := \xi(t_1; \xi_0, u)$, and

$$\Lambda := \int_0^{t_1} \ell(\xi(t; \xi_0, u), u, t)\ \mathrm{d}t \tag{3.5}$$

5: **else**

6: Return $\Lambda = \infty$

Figure 3.3: General form of the tile motion planning algorithm.

the vehicle state trajectory remain within the tile $\mathcal{R}$ at all times in the interval $(0, t_\mathrm{f})$, whereas the constraint (3.3) states the requirement that the position components leave the tile in a finite time t_f. The algorithm returns the time t_1 required to traverse the first cell of the tile $\mathcal{R}$, the time history $u_{[0,t_1]}$ over the interval $[0, t_1]$ of the control input u that enables traversal of the tile, the vehicle state ξ_1 at the boundary between the first and second cells of the tile $\mathcal{R}$ (see Fig. 3.4), and the cost Λ of traversal of the first cell as defined in (3.5).

Note that expression (3.2) does not depend explicitly on θ or ψ. In practice, θ or ψ may be subject to other constraints: for example, if $\psi = (\dot{x}, \dot{y})$ represents the vehicle's velocity, then $\|\psi(t)\|$ may be subject to lower and upper bounds. However,

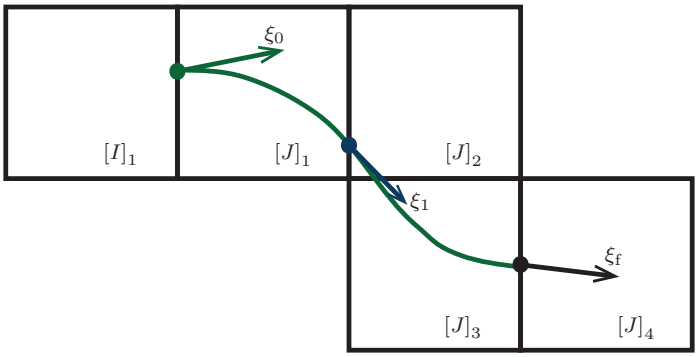


Figure 3.4: Illustration of tile motion planning for square cells, with $H = 3$.

these constraints are of no concern at the geometric planning level; their satisfaction will be ensured internally by TilePlan.

We discuss the implementation of TilePlan in the next chapter. Suppose, for now, that a tile motion planning algorithm satisfying the requirements in Fig. 3.4 is available. Figure 3.5 then describes the overall motion planning framework. It consists of a geometric path planner that repeatedly invokes TilePlan to determine H-costs of histories. The proposed motion planner associates with each node $I \in V_H$ (in addition to the label $d(I)$ and backpointer $b(I)$ of the standard label correcting algorithm) a vehicle state $\Xi(I) \in \mathcal{D}$, a time of traversal $\Theta(I) \in \mathbb{R}_+$, and an admissible control input $\Upsilon(I) \in \mathcal{U}_{\Theta(I)}$.

For the sake of clarity, the overall motion planning framework is described as a standard label correcting algorithm on the lifted graph. In practice, the motion planning framework may be implemented using the H-cost path planning algorithm described in Chapter 2.

Informally, the proposed planner searches for a path in the graph $\mathcal{G}_H$ by traversing one edge during each iteration, while simultaneously propagating the vehicle state forward. As previously mentioned, the choice of an appropriate control input for propagating the vehicle state are left to TilePlan.

The proposed planner produces a path $\Pi^* = (J_0, \ldots, J_P)$ in $\mathcal{G}_H$, where $[J_0]_1 = i_{\mathrm{S}}$ and $[J_P]_{H+1} = i_{\mathrm{G}}$. As discussed previously, Π^* corresponds to a sequence of cells, and

H-Cost Motion Planning Framework

Input: I_{S}, ξ_0, **Output:** $d, b, \Xi, \Theta, \Upsilon$

procedure INITIALIZE((I_{S}, ξ_0))

1: $\mathcal{P} \leftarrow \{I_{\mathrm{S}}\}$, $d(I_{\mathrm{S}}) \leftarrow 0$;
2: **for all** $I \in V_H \backslash \{I_{\mathrm{S}}\}$ **do**
3: $\quad d(I) = \infty$;
4: $\Xi(I) = \xi_0$, $\Theta(I) = 0$

procedure MAIN

1: INITIALIZE(I_{S}, ξ_0)
2: **while** $\mathcal{P} \neq \varnothing$ **do**
3: $\quad I \leftarrow$ REMOVE$(\mathcal{P})$
4: $\quad$ **for all** $J \in V_H$ such that $(I, J) \in E_H$ **do**
5: $\quad\quad \tau \leftarrow$ TILE$([I]_1, [J]_1^{H+1})$
6: $\quad\quad (t_1, u_{[0,t_1]}, \xi_1, \Lambda) \leftarrow$ TILEPLAN$(\mathcal{R}, \Xi(I))$
7: $\quad\quad$ **if** $d(I) + \Lambda < d(J)$ **then**
8: $\quad\quad\quad d(J) \leftarrow d(I) + \Lambda$
9: $\quad\quad\quad b(J) \leftarrow I$, $\Xi(J) \leftarrow \xi_1$
10: $\quad\quad\quad \Theta(J) \leftarrow t_1$, $\Upsilon(J) \leftarrow u_{[0,t_1]}$
11: $\quad\quad\quad$ INSERT$(\mathcal{P}, J)$

Figure 3.5: Pseudo-code for the overall motion planner.

the control input for traversing this sequence of cells is given by

$$u(t) := \Upsilon(J_k), \quad t \in \left[\sum_{m=1}^{k-1} \Theta(J_m), \sum_{m=1}^{k-1} \Theta(J_m) + \Theta(J_k) \right), \tag{3.6}$$

for each $k = 1, \ldots, P$.

In the next section, we show that, with increasing H, the costs of trajectories resulting from the proposed motion planner are non-increasing. An informal interpretation of this result is that as H is increased, the proposed motion planner erroneously rejects fewer admissible paths in $\mathcal{G}$ as infeasible. This result guides the selection of the value of H for implementing the proposed motion planner, in that it assures benefits (in terms of optimality of the resultant trajectory) in return for

expending computational resources for using larger values of H.

3.3 Dependence of Path Optimality on H

We assume here that the proposed motion planner solves *exactly* an H-cost shortest path problem, where the H-cost of an edge in $\mathcal{G}_H$ is determined by the tile motion planning algorithm. We discuss the variation of the minimum H-cost with respect to H via the following results. Here we denote by $\bar{P}$ the maximum number of nodes in any path in $\mathcal{G}$ from i_{S} to i_{G}.

Lemma 3.1. *Let $\pi = (j_0, \ldots, j_P)$ be an admissible path in $\mathcal{G}$. Then for each $H \in \mathbb{N}$,*

$$\tilde{\mathcal{J}}_{H+1}(\pi) \leqslant \tilde{\mathcal{J}}_H(\pi), \tag{3.7}$$

Proof. See Appendix B. □

Proposition 3.2. *Let $\mathcal{J}_H^*$ denote the minimum H-cost of paths in $\mathcal{G}$. Then $\{\mathcal{J}_H^*\}_{H=1}^{\bar{P}}$ is a non-increasing sequence.*

Proof. Let π be an admissible path in $\mathcal{G}$. By Lemma 3.1,

$$\tilde{\mathcal{J}}_{\bar{P}}(\pi) \leqslant \ldots \leqslant \tilde{\mathcal{J}}_1(\pi). \tag{3.8}$$

For $H \in \{1, \ldots, \bar{P}\}$, let π_H^* denote the H-cost shortest path in $\mathcal{G}$. Then for each admissible path π, $\mathcal{J}_H^* = \tilde{\mathcal{J}}_H(\pi_H^*) \leqslant \tilde{\mathcal{J}}_H(\pi)$ by optimality. In particular, for $\pi = \pi_{H-1}^*$,

$$\begin{aligned} \mathcal{J}_H^* = \tilde{\mathcal{J}}_H(\pi_H^*) &\leqslant \tilde{\mathcal{J}}_H(\pi_{H-1}^*) \\ &\leqslant \tilde{\mathcal{J}}_{H-1}(\pi_{H-1}^*) = \mathcal{J}_{H-1}^* \quad \text{(due to (3.8))}, \end{aligned}$$

and the result follows. □

Chapter 4

TilePlan via Model Predictive Control

The implementation of TilePlan is difficult mainly because (3.2) imposes a non-convex constraint on the state trajectory. To alleviate this difficulty, we take advantage of the fact that each cell in the sequence of cells associated with a tile is a convex region. We apply the concept of *effective target sets* introduced by Bertsekas and Rhodes [16], which enable the transformation of the constraint (3.2) to a convex constraint defined over a single cell.

The concept of effective target sets is described informally as follows. Consider a discrete-time dynamical system described by the difference equation

$$\xi(k+1) \;=\; f_{\mathrm{d}}(\xi(k), u(k)), \qquad k \in \mathbb{N}, \tag{4.1}$$

where $u(k) \in U$, for each $k \in \mathbb{N}$. Let $\xi_0 = \xi(0)$ be the initial state of the system, and let a horizon $N \in \mathbb{N}$ and a target set $\mathcal{X}_N \subseteq \mathcal{D}$ be pre-specified. Consider now the problem of finding an admissible control input sequence $u(0), \ldots, u(N-1)$ such that $\xi(N) \in \mathcal{X}_N$. To simplify this problem, consider the set $\mathcal{X}_{N-1} \subseteq \mathcal{D}$ defined by

$$\mathcal{X}_{N-1} \;:=\; \{\xi \in \mathcal{D} : \text{ there exists } u_{N-1} \in U \text{ such that } f_{\mathrm{d}}(\xi, u_{N-1}) \in \mathcal{X}_N\}.$$

Now suppose that there exists an admissible input sequence that solves the problem of driving the system state from $\xi(0) = \xi_0$ to $\xi(N) \in \mathcal{X}_N$. It follows that that $\xi(N-1) \in \mathcal{X}_{N-1}$. In other words, the original problem of finding a sequence of N admissible inputs can be reduced to the problem of finding a sequence of $N-1$ inputs with the constraint $\xi(N-1) \in \mathcal{X}_(N-1)$. Continuing the argument recursively, we may define sets $\mathcal{X}_k$ for $k = 1, \ldots, N-2$ by

$$\mathcal{X}_k \;:=\; \{\xi \in \mathcal{D} : \text{ there exists } u_k \in U \text{ such that } f_{\mathrm{d}}(\xi, u_k) \in \mathcal{X}_{k+1}\},$$

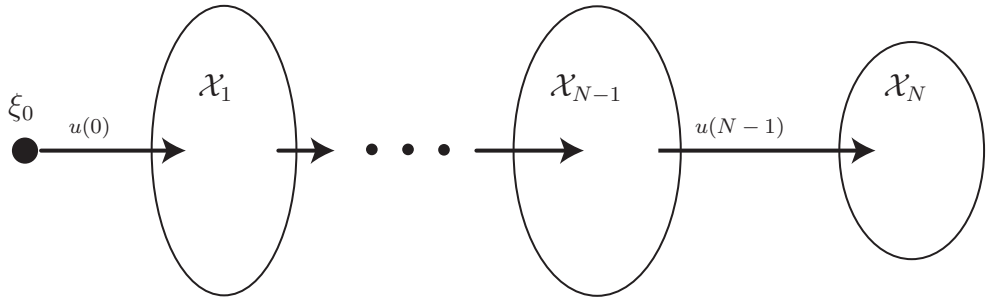


Figure 4.1: Illustration of the idea of effective target sets; arrows indicate a one-step evolution of (4.1).

and reduce the original problem of finding a sequence of N inputs to the problem of finding a single admissible input $u(0)$ such that $f(\xi(0), u(0)) \in \mathcal{X}_1$ (see Fig. 4.1).

4.1 Definitions of Effective Target Sets for TilePlan

Consider the tile associated with the H-history $(I, J) \in E_H$. We define a sequence $\{\mathcal{X}_k\}_{k=1}^{H+1}$ of subsets of the state space called the *effective target sets* as follows. Let

$$\mathcal{X}_H := ([J]_H \cap [J]_{H+1}) \times [-\pi, \pi] \times \mathcal{M}.$$

For each $k = 1, \ldots, H-1$, we define the effective target set $\mathcal{X}_k$ as the set of all states $\xi_k \in \mathcal{D}$ such that

$$\mathsf{x}(\xi_k) \in \mathsf{cell}([J]_k) \cap \mathsf{cell}([J]_{k+1}), \tag{4.2}$$

and such that there exists $t_{k+1} \in \mathbb{R}_+$ and an admissible control input $u_{k+1} \in \mathcal{U}_{t_{k+1}}$ such that the state trajectory $\xi(\cdot\,; \xi_k, u_{k+1})$ satisfies

$$\mathsf{x}(\xi(t; \xi_k, u_{k+1})) \in \mathsf{cell}([J]_{k+1}), \quad t \in (0, t_{k+1}), \tag{4.3}$$

$$\xi(t_{k+1}; \xi_k, u_{k+1}) \in \mathcal{X}_{k+1}. \tag{4.4}$$

The preceding definition of effective target sets allows a simplification of the tile motion planning problem as follows. Suppose there exist a time t_1 and a control $u_1 \in \mathcal{U}_{t_1}$ such that the resultant state trajectory $\xi(\cdot\,; \xi_0, u_1)$ satisfies

$$\mathsf{x}(\xi(t; \xi_0, u_1)) \in \mathsf{cell}([J]_1), \quad t \in (0, t_1), \tag{4.5}$$

$$\xi_1 := \xi(t_1; \xi_0, u_1) \in \mathcal{X}_1. \tag{4.6}$$

Note that, due to (4.2), the conditions (4.5)-(4.6) imply the satisfaction of (3.2)-(3.3) for $H = 1$. Next, because $\xi_1 \in \mathcal{X}_1$, it follows by (4.3)-(4.4) that there exists a $t_2 \in \mathbb{R}_+$ and an admissible control input $u_2 \in \mathcal{U}_{t_2}$ such that

$$\begin{aligned} \mathsf{x}(\xi(t; \xi_1, u_2)) &\in \mathsf{cell}([J]_2), \ t \in (0, t_2)\,, \\ \xi(t_2; \xi_1, u_2) &\in \mathcal{X}_2. \end{aligned}$$

In other words, the admissible control input u_{1-2} enables the vehicle's traversal through the cells corresponding to the vertices $[J]_1$ and $[J]_2$, where u_{1-2} is defined as the concatenation of the inputs u_1 and u_2 by

$$\begin{aligned} u_{1-2}(t) &:= \begin{cases} u_1(t), & t \in [0, t_1)\,, \\ u_2(t), & t \in [t_1, T_2]\,, \end{cases} \\ \text{with} \qquad T_2 &:= t_1 + t_2. \end{aligned}$$

It follows that

$$\mathsf{x}(\xi(t; \xi_0, u_{1-2})) \in \mathsf{cell}([J]_1) \cup \mathsf{cell}([J]_2), \ t \in (0, T_2)\,, \tag{4.7}$$

$$\xi(T_2; \xi_0, u_{1-2}) \in \mathcal{X}_2. \tag{4.8}$$

Due to (4.2), the conditions (4.7)-(4.8) imply the satisfaction of (3.2)-(3.3) for $H = 2$. Continuing recursively the preceding arguments, it follows that, for each $H \geqslant 2$, there exist $t_{k+1} \in \mathbb{R}_+$ and inputs $u_{k+1} \in \mathcal{U}_{t_{k+1}}$, for $k = 1, \ldots, H-1$, such that the admissible input u defined by

$$\begin{aligned} u(t) &:= \begin{cases} u_1(t), & t \in [0, T_1)\,, \\ u_2(t), & t \in [T_1, T_2)\,, \\ \vdots & \vdots \\ u_H(t), & t \in [T_{H-1}, T_H]\,, \end{cases} \\ \text{with} \qquad T_k &:= \textstyle\sum_{m=1}^{k} t_m, \end{aligned} \tag{4.9}$$

solves the tile motion planning problem.

Thus, if the effective target sets $\mathcal{X}_k$, the corresponding times of traversal t_{k+1} and the control inputs u_{k+1} in (4.9) are known for each $k = 1, \ldots, H-1$, then the tile motion planning problem is equivalent to the problem of finding u_1 and t_1 as described above. Crucially, (4.5) constrains the position components of the state trajectory to lie within a convex set. Furthermore, we may replace $\mathcal{X}_1$ in (4.6) by an interior convex approximating set $\tilde{\mathcal{X}}_1 \subset \mathcal{X}_1$ thus transforming the tile motion planning problem into the problem of finding u_1 and t_1 subject to convex constraints.

4.2 MPC Problem Formulation

In the MPC formulation of the tile motion planning problem, we consider a linear approximation to the vehicle dynamical model as

$$\dot{\xi} \;=\; A\xi + B_1 u + B_2, \tag{4.10}$$

where

$$A := \left.\frac{\partial f}{\partial \xi}\right|_{(\xi_0,u_0)}, \quad B_1 := \left.\frac{\partial f}{\partial u}\right|_{(\xi_0,u_0)},$$

and $B_2 := f(\xi_0, u_0) - A\xi_0 - B_1 u_0$. A discrete-time approximation of (4.10), with a sampling period t_s, is given by

$$\xi(k+1) \;=\; A_\mathrm{d}\xi(k) + B_{1\mathrm{d}} u(k) + B_{2\mathrm{d}},$$

where $A_\mathrm{d} := e^{At_\mathrm{s}}$, $B_{i\mathrm{d}} := \int_0^{t_\mathrm{s}} e^{A(t_\mathrm{s}-\tau)} B_i d\tau$, for $i = 1, 2$.

We denote by H_P the prediction horizon, by $\tilde{\ell} : \mathcal{D} \times U \to \mathbb{R}_+$ a pre-specified incremental cost function, and by $\tilde{\Lambda}_\mathrm{f} : \mathcal{D} \to \mathbb{R}_+$ a pre-specified terminal cost function. The MPC problem is then described as follows:

$$\begin{aligned} &\min_{H_\mathrm{P}\in\mathbb{N},(u(0),\ldots,u(H_\mathrm{P}))} \left\{ \tilde{\Lambda}_\mathrm{f}(\xi(H_\mathrm{P})) + \sum_{k=0}^{H_\mathrm{P}-1} \tilde{\ell}(\xi(k), u(k)) \right\}, \\ &\text{subject to } \xi(H_\mathrm{P}) \in \tilde{\mathcal{X}}_1, \qquad \xi(k) \in \mathsf{cell}([J]_1), \\ &\text{and } u(k) \in U, \text{ for each } k \in \{0, \ldots, H_\mathrm{P}-1\}. \end{aligned} \tag{4.11}$$

Note that the incremental cost $\tilde{\ell}$ in (4.11) need not be the same as the incremental cost ℓ in (3.5): the role of TilePlan in the overall motion planning framework is that of ensuring *feasibility* of traversal of tiles, whereas it is the higher-level discrete planner that searches for an optimal sequence of cell transitions. To solve the tile motion planning problem, the MPC-problem (4.11) is solved; the first input of the resulting input sequence is chosen and a new state is obtained by integrating over time t_s the actual (nonlinear) vehicle model; the linearization (4.10) is performed about the new state [105]; and the preceding steps are repeated.

4.3 Computation of Effective Target Sets

As discussed in the preceding section, effective target sets simplify the MPC implementation of TilePlan. The computation of the effective target sets themselves, however, is challenging for general nonlinear dynamical systems.

In light of the fact that the vehicle state includes the configuration (x, y, θ), we consider first the computation of the intersections of the effective target sets with the configuration space $\mathcal{C} = \mathbb{R}^2 \times \mathbb{S}^1$. To this end, we define the *effective target configuration sets* by $\mathcal{C}_k := \mathcal{X}_k \cap \mathcal{C}$, and, in what follows, we outline a geometric scheme of computing the sets $\mathcal{C}_k$.

Assumption 4.1. *The geometric curves in the plane that can be feasibly traversed by the vehicle satisfy a local upper bound on their curvatures.*

To justify this Assumption, we characterize as follows the curvature of the geometric paths corresponding to projections on $\mathbb{R}^2$ of feasible state trajectories. Note that the following kinematical equations relate the inertial position coordinates x, y to the orientation θ irrespective of the vehicle dynamical model:

$$\dot{x}(t) = v(t)\cos\theta(t), \quad \dot{y}(t) = v(t)\sin\theta(t). \tag{4.12}$$

The curvature of the planar curve $p(t) = (x(t), y(t))$ is [86]:

$$\kappa(t) \;=\; \sqrt{\frac{\langle \dot{p}, \dot{p}\rangle\langle \ddot{p}, \ddot{p}\rangle - \langle \dot{p}, \ddot{p}\rangle^2}{\langle \dot{p}, \dot{p}\rangle^3}} = \left|\frac{\dot{\theta}}{v}\right|, \tag{4.13}$$

by (4.12). In the context of the vehicle dynamical model, the curvature of feasible paths is related to the set of admissible control values via the term in the numerator of (4.13), i.e.,

$$\min_{u(t)\in U}\left|\frac{\dot{\theta}(\xi(t), u(t))}{v(t)}\right| \leqslant \kappa(t) \leqslant \max_{u(t)\in U}\left|\frac{\dot{\theta}(\xi(t), u(t))}{v(t)}\right|.$$

The upper bound $\kappa^{\max}$ on the curvature of a feasible path over a given time interval of interest $[0, t_{\mathrm{f}}]$ is $\min_{t\in[0,t_{\mathrm{f}}]}\kappa(t)$, and it follows that

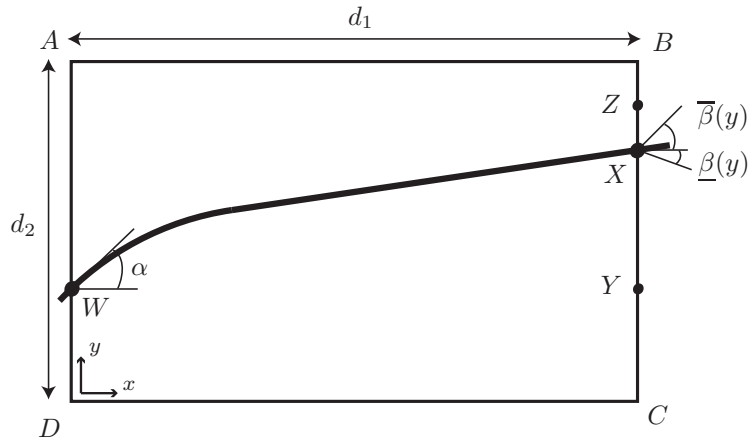
$$\kappa^{\max} \leqslant \min_{t\in[0,t_{\mathrm{f}}]}\max_{u(t)\in U}\left|\frac{\dot{\theta}(\xi(t), u(t))}{v(t)}\right|. \tag{4.14}$$

In light of Assumption 4.1, we, compute the sets $\mathcal{C}_k$ by solving the following problems in plane geometry.

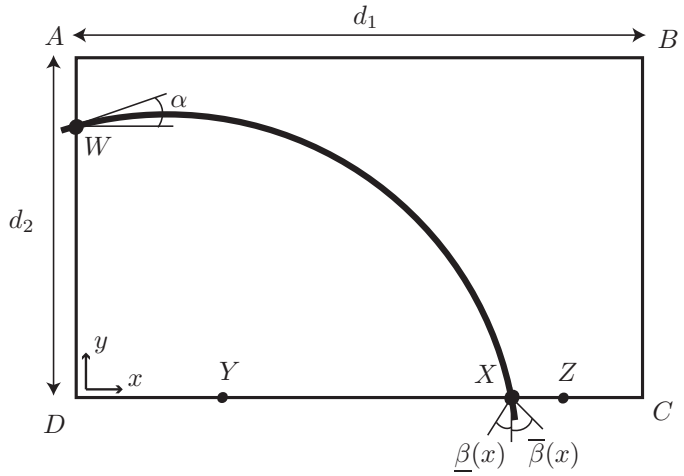
Let $ABCD$ be a rectangle. We attach a Cartesian axes system as shown in Fig. 4.2. Let the dimensions of the rectangle be d_1 and d_2, and let $r > 0$ be fixed.

Definition 4.2 (Type 1 Admissible Path)**.** *Let $\underline{\beta}(x), \overline{\beta}(x)$, $x \in [0, d_2]$ be functions such that $-\frac{\pi}{2} \leqslant \underline{\beta}(x) \leqslant \overline{\beta}(x) \leqslant \frac{\pi}{2}$. Let $Y = (d_1, y), Z = (d_1, z)$ be points on the segment BC with $y \leqslant z$. A path Γ is a Type 1 admissible path if it satisfies the following properties:*

1. *(Curvature Boundedness): The curvature at any point on Γ is at most r^{-1},*
2. *(Containment): Γ intersects the segment BC in exactly one point $X = (d_1, x)$ such that $x \in [y, z]$, and it may intersect segment AB and/or segment CD in at most one point each, and*
3. *(Terminal Orientation): $\Gamma'(X) \in \left[\underline{\beta}(x), \overline{\beta}(x)\right]$.*



(a) Type 1 admissible path.



(b) Type 2 admissible path.

Figure 4.2: Setup for Problems 4.4 and 4.5, which is used in the computation of effective target configuration sets.

A Type 2 admissible path is defined analogously for traversal across adjacent edges.

Definition 4.3 (Type 2 Admissible Path)**.** *Let $\underline{\beta}(x), \overline{\beta}(x)$, $x \in [0, d_1]$ be functions such that $-\frac{\pi}{2} \leqslant \underline{\beta}(x) \leqslant \overline{\beta}(x) \leqslant \frac{\pi}{2}$. Let $Y = (y, 0), Z = (z, 0)$ be points on the segment CD with $y \leqslant z$. A path Γ is a Type 2 admissible path if it satisfies the following properties:*

1. *(Curvature Boundedness): The curvature at any point on Γ is at most r^{-1},*

2. *(Containment): Γ intersects the segment CD in exactly one point $X = (x, 0)$*

such that $x \in [y, z]$, and it may intersect segment AB and/or segment BC in at most one point each, and

3. *(Terminal Orientation):* $\Gamma'(X) + \frac{\pi}{2} \in \left[\underline{\beta}(x), \overline{\beta}(x)\right]$.

We state two geometric problems as follows. Let $\underline{\beta}$, $\overline{\beta}$, Y, and Z be as in the preceding definitions. Let $W = (0, w)$ and $r > 0$ be fixed.

Problem 4.4 (*Traversal across parallel edges*)**.** *Find bounds $\underline{\alpha}, \overline{\alpha}$ such that for all $\alpha \in [\underline{\alpha}, \overline{\alpha}]$, there exists a Type 1 admissible path Γ with $\Gamma(0) = W$ and $\Gamma'(W) = \alpha$.*

Problem 4.5 (*Traversal across adjacent edges*)**.** *Find bounds $\underline{\alpha}, \overline{\alpha}$ such that for all $\alpha \in [\underline{\alpha}, \overline{\alpha}]$, there exists a Type 2 admissible path Γ with $\Gamma(0) = W$ and $\Gamma'(W) = \alpha$.*

Problems 4.4 and 4.5 appear in the recursive computation of effective target configurations as follows. Suppose that the effective target configuration set $\mathcal{C}_{k+1}$ is known, for $k \in \{1, \ldots, H-1\}$. Then we may express $\mathcal{C}_{k+1}$ as the product set of a line segment on the boundary between cells $\mathsf{cell}([J]_{k+1})$ and $\mathsf{cell}([J]_{k+2})$ with an interval of allowable orientations on this line segment. In other words, we may express $\mathcal{C}_{k+1}$ in terms of the points Y, Z and the functions $\underline{\beta}, \overline{\beta}$ used in Definitions 4.2 and 4.3. We may then solve Problem 4.4 or 4.5, as applicable for the cell $\mathsf{cell}([J]_{k+1})$, for each point on the line segment forming the boundary between cells $\mathsf{cell}([J]_k)$ and $\mathsf{cell}([J]_{k+1})$ and obtain allowable orientations for each point on this line segment. The product set of these allowable orientations and this line segment is precisely the set $\mathcal{C}_k$. The effective target configuration sets may thus be computed recursively by repeatedly solving Problems 4.4 and 4.5 as applicable for each cell, with $\mathcal{C}_H := (\mathsf{cell}([J]_H) \cap \mathsf{cell}([J]_{H+1})) \times \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$.

The solutions to Problems 4.4 and 4.5 are discussed Chapter 5 and in Appendix C; first, we discuss the characterization of the curvature constraints on feasible paths for different vehicle models.

4.4 Illustrative Examples

We illustrate the computation of the curvature constraints and of the effective target sets for three vehicle models: the Dubins car kinematic model, a particle dynamical model with "friction ellipse" type input constraints, and an aircraft point-mass navigational model.

4.4.1 Dubins Car Model

The Dubins car kinematic model is described by

$$\dot{x}(t) = v\cos\theta(t), \qquad \dot{y}(t) = v\sin\theta(t), \qquad \dot{\theta}(t) = u(t),$$

where x, y, and θ are, respectively, the position coordinates and the orientation of the vehicle with respect to a pre-specified inertial axes system; $v > 0$ is the (fixed) forward speed of the vehicle; and u is the steering control input. The set of admissible control inputs is $U := [-1/r, 1/r]$, for a pre-specified $r > 0$.

By (4.14), it follows that

$$\kappa^{\max} \leqslant \min_{t\in[0,t_{\mathrm{f}}]} \max_{u(t)\in U} \left|\frac{u(t)}{v(t)}\right| = \left|\frac{1/r}{v}\right| = (rv)^{-1}.$$

For the Dubins car model, the configuration space $\mathcal{C}$ and the state space $\mathcal{D}$ are identical, and hence the effective target sets coincide with the effective target configurations sets.

4.4.2 Particle Dynamical Model

We consider a vehicle dynamical model described by

$$\begin{aligned} &\dot{x}(t) = v(t)\cos\theta(t), \quad \dot{y}(t) = v(t)\sin\theta(t), \\ &\dot{\theta}(t) = u_2(t), \qquad\qquad \dot{v}(t) = u_1(t), \end{aligned}$$

where $v > 0$ is the forward speed of the vehicle; u_1 is the acceleration input, and u_2 is the steering input. The speed v is constrained to lie within pre-specified bounds

$v_{\min}$ and $v_{\max}$; these bounds may be different for different regions of the workspace. The set of admissible control inputs is

$$U \;:=\; \{(a,\omega) : \left(\frac{v\omega}{f_{\mathrm{r}}^{\max}}\right)^2 + \left(\frac{a}{f_{\mathrm{t}}^{\max}}\right)^2 \leqslant 1\}, \tag{4.15}$$

where $f_{\mathrm{r}}^{\max}$ and $f_{\mathrm{t}}^{\max}$ are pre-specified. The input constraint defined by (4.15) is an example of a "friction ellipse" constraint that models the limited tire frictional forces available for acceleration and steering of the vehicle. Finally, we denote by $v_j^{\max}$ and $v_j^{\min}$ pre-specified bounds on the vehicle speed inside the cell corresponding to the vertex $j \in V$.

We may now compute the effective target sets for this vehicle model as follows. We may assume, as in [151], that the input constraint (4.15) is tightened, as in (4.16) below, to ensure controllability of the vehicle at all times:

$$U \;:=\; \{(a,\omega) : \left(\frac{v\omega}{f_{\mathrm{r}}^{\max}}\right)^2 + \left(\frac{a}{f_{\mathrm{t}}^{\max}}\right)^2 \leqslant 1 - \varepsilon^2\}, \tag{4.16}$$

where $\varepsilon \ll 1$. The constraint (4.16) implies that acceleration/deceleration of the vehicle with $|\dot{v}| \geqslant \varepsilon f_{\mathrm{t}}^{\max}$ is always feasible. Also note that for cells involving traversal across parallel edges, the vehicle traverses at least a distance d.

Let $\mathcal{X}_H := \mathsf{cell}([J]_H) \cap \mathsf{cell}([J]_{H+1}) \times [-\pi,\pi] \times [\underline{v}_H, \overline{v}_H]$, where $\overline{v}_H := v_{j_{H+1}}^{\max}$ and $\underline{v}_H := v_{j_{H+1}}^{\min}$, where $j_k := [J]_k$ for $k \in \{1,\ldots,H+1\}$. It follows by the arguments in the preceding paragraph that upper and lower bounds for the vehicle speed v at each of the boundaries of adjacent cells in the tile are given by

$$\begin{aligned}
\overline{v}_k &= \min\{v_{j_k}^{\max}, v_{j_{k+1}}^{\max}, \sqrt{\overline{v}_{k+1} + 2\varepsilon f_{\mathrm{t}}^{\max} d}\},\\
\underline{v}_k &= \max\{v_{j_k}^{\min}, v_{j_{k+1}}^{\min}, \sqrt{\underline{v}_{k+1} - 2\varepsilon f_{\mathrm{t}}^{\max} d}\},
\end{aligned}$$

whenever the cell corresponding to j_k involves traversal across parallel edges, and by

$$\overline{v}_k = \min\{v_{j_k}^{\max}, \overline{v}_{k+1}\}, \qquad \underline{v}_k = \min\{v_{j_k}^{\min}, \underline{v}_{k+1}\},$$

whenever the cell corresponding to $j_k \in V$ involves traversal across adjacent edges. The upper bound $\kappa_k^{\max}$ on the curvature of paths traversing the cell corresponding to

$j_k \in V$ for $k = 1, \ldots, H-1$, is, by (4.14),

$$\kappa_k^{\max} = \frac{f_{\mathrm{r}}^{\max}\sqrt{1-\varepsilon^2}}{(\max\{\overline{v}_k, \overline{v}_{k+1}\})^2}. \tag{4.17}$$

The bound (4.17) on the curvature of feasible paths is conservative because the bound on the vehicle speed in the denominator does not involve the initial speed v_0, i.e., the maximum reachable speed within each of the cells in the tile may be lower than $\max\{\overline{v}_k, \overline{v}_{k+1}\}$, and may be a less conservative bound on the speed (and consequently, on the curvature). The exact calculation of the maximum reachable speed involves the (numerically intensive) solution of an optimal control problem; however, an easily computable, heuristic approximation may be obtained by considering maximum acceleration along the longest linear path within the cell (i.e., the diagonal of length $\sqrt{2}d$). Thus, a less conservative, heuristic bound on the curvature is given by

$$\kappa_k^{\max} = \frac{f_{\mathrm{r}}^{\max}\sqrt{1-\varepsilon^2}}{(\min\{\max\{\overline{v}_k, \overline{v}_{k+1}\}, \sqrt{v_0^2 + 2\sqrt{2}f_{\mathrm{t}}^{\max}d}\})^2}.$$

4.4.3 Aircraft Navigational Model

We consider a point-mass aircraft navigational model described by

$$\begin{aligned}
\dot{x}(t) &= v(t)\cos\gamma(t)\cos\psi(t),\\
\dot{y}(t) &= v(t)\cos\gamma(t)\sin\psi(t),\\
\dot{z}(t) &= v(t)\sin\gamma(t),\\
\dot{\psi}(t) &= -\frac{q(t)C_{\mathrm{L}}(t)}{mv(t)\cos\gamma(t)},\\
\dot{v}(t) &= \left(T(t) - q(v(t))C_{\mathrm{D},0} - KC_{\mathrm{L}}^2(t)\right)/m,\\
\dot{\gamma}(t) &= \frac{1}{mv(t)}\left(q(v(t))C_{\mathrm{L}}(t)\cos\phi(t) - mg\cos\gamma(t)\right),
\end{aligned}$$

where x, y, and z denote the inertial position coordinates, v denotes the speed, ψ denotes the aircraft heading, γ denotes the flight path angle, $q(v) := \frac{1}{2}\rho v^2 S$ denotes the dynamic pressure, m denotes the mass of the aircraft, and $C_{\mathrm{D},0}$ and K

are pre-specified constants. The control inputs are the thrust $T \in [T_{\min}, T_{\max}]$, the lift coefficient $C_{\mathrm{L}} \in [C_{\mathrm{L,min}}, C_{\mathrm{L,max}}]$, and the bank angle $\phi \in [\phi_{\min}, \phi_{\max}]$, where the bounds for the admissible control inputs are pre-specified.

We consider the motion of the aircraft in the horizontal plane, i.e., $\gamma(t) = 0$ and $\dot{\gamma}(t) = 0$, and to this end we set[1]

$$C_{\mathrm{L}}(t) = mg/(q(v(t))\cos\phi(t)).$$

We may assume the aircraft's cruise speed to be a constant v_{cr}. The thrust input is then given by

$$\begin{aligned} T(v_{\mathrm{cr}}, \phi(t)) &= q(v_{\mathrm{cr}})C_{\mathrm{D},0} - KC_{\mathrm{L}}^2 \\ &= q(v_{\mathrm{cr}})C_{\mathrm{D},0} - \frac{K(mg)^2}{\left(q(v_{\mathrm{cr}})\right)^2 \cos^2\phi(t)}. \end{aligned}$$

Alternatively, we may assume a constant thrust input of value $T(v_{\mathrm{cr}}, 0)$, and allow small decreases in the aircraft speed during turning flight. In either case, the upper bound on the curvature, by (4.14), is given by $\kappa_k^{\max} = g\tan(\min|\phi_{\min}|, |\phi_{\max}|)/v_{\mathrm{cr}}$, for $k = 1, \ldots, H-1$.

[1]We assume $C_{\mathrm{L,min}} \leqslant \frac{mg}{q} \leqslant \frac{mg}{q\cos\phi_{\max}} \leqslant C_{\mathrm{L,max}}$, which physically implies that the lift coefficient can be sufficiently varied to allow the vertical component of the lift to equal the aircraft's weight at any bank angle.

Chapter 5

Curvature-bounded Paths inside Rectangular Channels: Special Case

In this chapter, we discuss the constructions of the effective target configuration sets introduced in the previous chapter. These constructions directly lend themselves to a problem of independent interest, namely, that of determining the existence of curvature-bounded paths traversing rectangular channels (i.e., a channel consisting of a sequence of rectangular cells). In what follows, we first introduce this problem of curvature-bounded traversal of rectangular channels, we discuss its relation to the construction of effective target configuration sets, and we discuss its solution by recursively solving Problems 4.4 and 4.5 previously introduced. Finally, we will outline the solutions of Problems 4.4 and 4.5 themselves. In this chapter, we provide a computational procedure for solving these problems for the special case of square cells with the upper bound on the curvature sufficiently small relative to the dimensions of the square; in Appendix C, we treat the general case of rectangular cells of arbitrary dimensions with no assumptions on the curvature bound relative to the rectangle dimensions.

We begin by introducing some terminology that will be used in the rest of this chapter and in Appendix C. A *path between points* W and X in the plane is a continuously differentiable curve $\Gamma := \{s \mapsto (x(s), y(s)) \in \mathbb{R}^2 : 0 \leqslant s \leqslant 1\}$ such that $W = (x(0), y(0))$ and $X = (x(1), y(1))$. We will denote by $\Gamma(w)$ the point $(x(w), y(w)) \in \mathbb{R}^2$ on the path Γ, and by $\Gamma'(W)$ the angle of the tangent to Γ at the point $W = (x(w), y(w))$ for all $w \in [0, 1]$, i.e.,

$$\Gamma'(W) := \tan^{-1}\left(\frac{\mathrm{d}y/\mathrm{d}s}{\mathrm{d}x/\mathrm{d}s}\right)\Bigg|_{s=w}.$$

Also, we will denote by (W, α) the configuration in $\mathcal{C}$ specified by the position $W \in \mathbb{R}^2$ and the orientation $\alpha \in [-\pi, \pi]$. A path *between a configuration* (W, α) *and a point* X is a path Γ between the points W and X satisfying $\Gamma'(W) = \alpha$. Similarly, a path *between two configurations* (W, α) and (X, β) is a path Γ between the points W and X satisfying $\Gamma'(W) = \alpha$ and $\Gamma'(X) = \beta$.

Definition 5.1 (Rectangular channel)**.** *A rectangular channel* $\bar{\mathcal{R}}^C$ *is a finite sequence of rectangles* $\{R_n\}_{n=1}^C$, $C \in \mathbb{N}$, *with disjoint interiors, such that*

(i) For each $n \in \{1, \ldots, C-1\}$, *exactly one edge of* R_n *has a non-empty intersection with exactly one edge of* R_{n+1}, ,

(ii) For all $m, n \in \{1, \ldots, C\}$, *the edges of* R_n *and* R_m *do not intersect or overlap whenever* $m \notin \{n-1, n, n+1\}$.

Problem 5.2 (Curvature-bounded Traversal of Rectangular Channels)**.** *Let* $\bar{\mathcal{R}}^C$ *be a rectangular channel, and let* W *be a point on any one of the three edges of* R_1 *that do not intersect* R_2. *Let* $\alpha \in [-\Gamma, \Gamma]$ *be a specified angle. For any set of positive real numbers* $r_n > 0$, $n = 1, \ldots, C$, *determine if there exists a path* Γ *such that:*

(i) $\Gamma(0) = W$ *and* $\Gamma'(W) = \alpha$,

(ii) The point $X := \Gamma(1)$ *lies on an edge of the rectangle* R_C *(pre-specified from among the three edges of* R_C *that do not overlap with any edge of* R_{C-1}*), and* $\Gamma'(X)$ *lies in a specified set of allowable terminal tangent angles,*

(iii) The path Γ *does not leave* $\bar{\mathcal{R}}^C$, *i.e.* $(x(s), y(s)) \in \cup_{n=1}^C R_n$ *for every* $s \in [0, 1]$,

(iv) For each $n = 1, \ldots, C$, *the curvature of* Γ *at any point in rectangle* R_n *is at most* r_n^{-1}.

Problem 5.2 can be solved using the recursive procedure outlined in Section 4.3, which we discuss in detail in the next section. In the context of uniform square cell

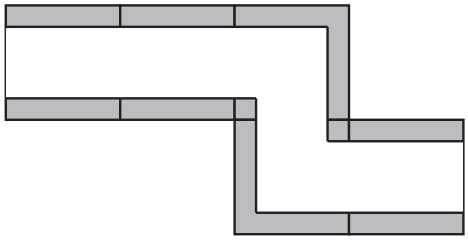
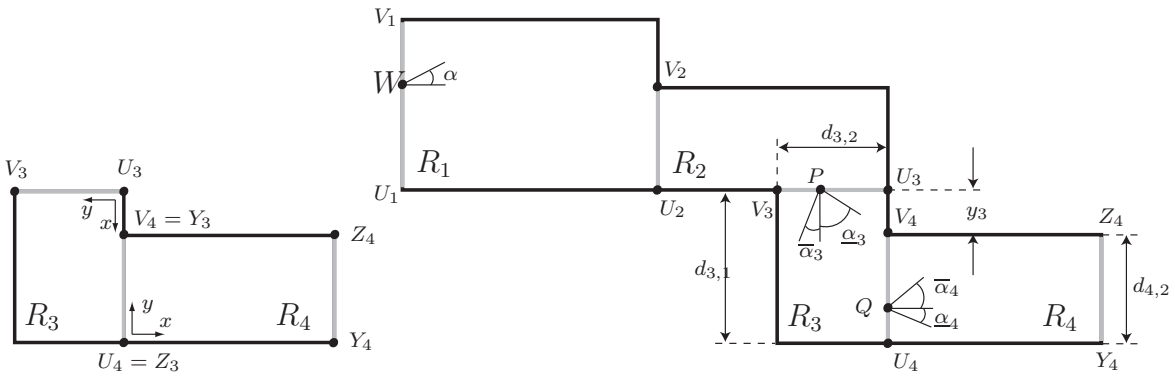


Figure 5.1: Shrunken channel within a tile, to incorporate the vehicle's finite size into the motion planning algorithm.

decompositions, each tile corresponds to a channel consisting of square cells of equal dimensions. However, in Problem 5.2, we allow rectangles of arbitrary dimensions for three main reasons: (1) the problem of curvature-bounded traversal in constrained environments is of independent interest, and it has been studied in several works in the literature (cf. [2, 14, 18, 44] and references therein), (2) channels consisting of squares of different dimensions arise in multi-resolution cell decompositions (as discussed in Chapter 6), and (3) to incorporate the finite size of vehicles in motion planning, the traversal of tiles may constrained to allow paths only within a shrunken channel lying within the original channel (see Fig. 5.1). In this case, the new channel can be decomposed into rectangles, but not necessarily squares.

5.1 Recursive Constructions of Effective Target Configuration Sets

We attach a coordinate axes system to each rectangle of $\bar{\mathcal{R}}^C$ in a manner consistent with the axes system used in the statement of Problems 4.4 and 4.5 (see Fig. 5.2). The dimensions of each rectangle along the x and y axes are denoted, respectively, as $d_{n,1}$ and $d_{n,2}$. For example, the axes system attached to the rectangle R_3 in Fig. 5.2 has its origin at the point U_3, with the positive x-axis along the segment U_3Y_3, and the positive y-axis along the segment U_3V_3. We denote the length of the rectangle R_n (dimension along the local x-axis) by $d_{n,1}$ and the height of the rectangle (dimension along the local y-axis) by $d_{n,2}$. We may identify rigid geometric transformations



(a) Notation used in Section 5.1 (b) Illustrative example for the proposed algorithm.

Figure 5.2: Supporting figures for Section 5.1.

(i.e., a sequence of rotations and reflections) that align the entry and exit segments of R_n to the segments AD and BC, respectively, for traversal across parallel edges, or to the segments AD and CD, respectively, for traversal across adjacent edges. We will denote by ϱ_n the minimum number of reflections involved in the transformation associated with the rectangle R_n.

For each rectangle R_n, $n = 2, 3, \ldots, C-1$, we refer to the segments formed by the intersections $R_{n-1} \cap R_n$ and $R_n \cap R_{n+1}$, respectively, as the *entry and exit segments*. For the rectangle R_1 (resp. rectangle R_C), the entry segment (resp. exit segment) is specified arbitrarily; however, the entry segment of R_1 (resp. exit segment R_C) must lie on an edge of R_1 (resp. R_C distinct from the edge intersecting R_2 (resp. R_{n-1}). We denote the endpoints of the entry segment by U_n and V_n, and the endpoints of the exit segment by Y_n and Z_n. We specify the coordinates of the points U_n, V_n, Y_n, Z_n, by the corresponding lower case letters, i.e., $V_n = (0, v_n)$, etc.

Recall from Section 4.3 that the effective target configuration sets are the product sets of line segments on the boundaries between successive cells with intervals of allowable orientations. In the context of the terminology introduced in this chapter, the effective target configuration sets are the product sets of segments lying on the exit segments of each rectangle with intervals of allowable orientations. We outline in what follows the computation of the upper and lower bounds of these intervals.

Recursive Computations of Effective Target Configuration Sets

Input: Rectangular channel $\bar{\mathcal{R}}^C$, **Output:** $\underline{\alpha}_n, \overline{\alpha}_n$, for each $n \in \{1, \ldots, C\}$

1: $\overline{\alpha}_{C+1}(q) \leftarrow \frac{\pi}{2}$ and $\underline{\alpha}_{C+1}(q) \leftarrow -\frac{\pi}{2}$,
for all $q \in [0, d_{C,2}]$ if R_C involves traversal across parallel edges, or otherwise,
for all $q \in [0, d_{C,1}]$ if R_C involves traversal across adjacent edges.
2: $\varrho_{C+1} \leftarrow 0$
3: **for** $n = C, C-1, \ldots, 1$ **do**
4: **if** $\varrho_n + \varrho_{n-1}$ is odd **then**
5: $\underline{\beta}_n(q) \leftarrow -\overline{\alpha}_{n+1}\left(y_n - (p - v_{n+1})\right)$, for all $p \in [u_{n+1}, v_{n+1}]$, $q \in [y_n, z_n]$
6: $\overline{\beta}_n(q) \leftarrow -\underline{\alpha}_{n+1}\left(y_n - (p - v_{n+1})\right)$, for all $p \in [u_{n+1}, v_{n+1}]$, $q \in [y_n, z_n]$
7: **else**
8: $\underline{\beta}_n(q) \leftarrow \underline{\alpha}_{n+1}(p)$, for all $p \in [u_{n+1}, v_{n+1}]$, $q \in [y_n, z_n]$
9: $\overline{\beta}_n(q) \leftarrow \overline{\alpha}_{n+1}(p)$, for all $p \in [u_{n+1}, v_{n+1}]$, $q \in [y_n, z_n]$
10: Find the values of $\underline{\alpha}_n(q)$ and $\overline{\alpha}_n(q)$ for all $q \in [u_n, v_n]$ by solving Problem 4.4 or Problem 4.5 for rectangle R_{n-1}, as applicable.

Figure 5.3: Pseudo-code of the recursive procedure for computing the effective target configuration sets.

For every point $Q = (q, 0)$ (or $Q = (d_{n,1}, q)$, as applicable), on the segment $Y_n Z_n$, $n = 1, \ldots, C$, we denote by $\underline{\beta}_n(q)$ and $\overline{\beta}_n(q)$, respectively, the lower and upper bounds of the allowable terminal orientations for $q \in [y_n, z_n]$. Similarly, for every point $P = (0, p)$ on the segment $U_n V_n$, $n = 1, \ldots, C$, we denote by $\underline{\alpha}_n(p)$ and $\overline{\alpha}_n(p)$, respectively, the lower and upper bounds resulting from the solution of Problem 4.4 (or Problem 4.5, as applicable), for $p \in [u_n, v_n]$. Note that $[\underline{\alpha}_n(p), \overline{\alpha}_n(p)]$ is the interval of allowable orientations at point P in the effective target configuration set of the rectangle R_{n-1}. The angles $\underline{\alpha}_n(\cdot), \overline{\alpha}_n(\cdot), \underline{\beta}_n(\cdot)$, and $\overline{\beta}_n(\cdot)$ are all measured with respect tothe local coordinate axes system attached to R_n.

The recursive algorithm for constructing the effective target configuration sets, i.e., for computing the bounds on the intervals of allowable orientations at the boundaries of successive cells, is provided in Fig. 5.3. To better explain the notation introduced

in this section and the algorithm described in Fig. 5.3, we illustrate the execution of this algorithm using an example.

5.1.1 Illustrative Example

Let $\bar{\mathcal{R}}^4 = \{R_n\}_{n=1}^4$ be a rectangular channel with four rectangles, as shown in Fig. 5.2(b), and let $r_n > 0$, $n = 1, \ldots, 4$, be given. The points U_n, V_n, $n = 1, \ldots, 4$ and the points Y_4, Z_4 are shown in Fig. 5.2(b). We note that $Y_1 = U_2$, $Z_1 = V_2$, $Y_2 = V_3$, $Z_2 = U_3$, $Y_3 = V_4$, and $Z_3 = U_4$. Given a prescribed initial entry point W on the segment U_1V_1 and given a prescribed initial tangent angle $\alpha \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$, we wish to determine if there exists a path satisfying the conditions described in Problem 5.2.

Following the algorithm stated above, we note that the last rectangle in the channel, i.e., R_4, involves traversal across parallel edges, and by Line 1, we initialize $\overline{\alpha}_5$ and $\underline{\alpha}_5$ as

$$\overline{\alpha}_5(q) = \frac{\pi}{2}, \quad \underline{\alpha}_5(q) = -\frac{\pi}{2}, \qquad q \in [0, d_{4,2}]\,.$$

Next, we note that the entry and exit segments of rectangle R_4 are aligned, respectively, with segments AD and BC of Fig. 4.2. Thus, the number of reflections ϱ_4 occurring in the transformation required for R_4 is zero, and by the condition in Line 4, we set

$$\overline{\beta}_4 = \overline{\alpha}_5 = \frac{\pi}{2}, \quad \underline{\beta}_4 = \underline{\alpha}_5 = -\frac{\pi}{2}.$$

Next, by Line 10, we solve Problem 4.4 for each point $Q = (0, q)$, $q \in [0, d_{4,2}]$ on the segment U_4V_4, and we obtain the values taken by the functions $\underline{\alpha}_4(q)$ and $\overline{\alpha}_4(q)$. Repeating Lines 4-9 for $n = C - 1 = 3$, we first note that tectangle R_3 involves traversal across adjacent edges, and the entry and exit segments of R_3 may be aligned with segments AD and DC of Fig 4.2 after a reflection about an axis parallel to the segment U_4V_4, followed by a rotation through $\frac{\pi}{2}$ rad. Thus, $\varrho_3 = 1$, and $\varrho_3 + \varrho_4 =$

$1 + 0 = 1$, and by the condition in Line 4, we set

$$\begin{aligned} \overline{\beta}_3(q) &= -\underline{\alpha}_4(y_3 - (q - v_4)), \\ \underline{\beta}_3(q) &= -\overline{\alpha}_4(z_3 - (q - u_4)), \quad q \in [y_3, z_3], \end{aligned}$$

where $z_3 = d_{3,1}$, $y_3 = \ell\,(U_3V_4)$, $v_4 = d_{4,2}$, and $u_4 = 0$ (see Fig. 5.2(b)). By Line 10, we solve Problem 4.5 for each point $P = (0, p)$, $p \in [0, d_{3,2}]$ on the segment U_3V_3 to obtain values taken by the functions $\underline{\alpha}_3(p)$ and $\overline{\alpha}_3(p)$. Proceeding further in a similar manner, we may obtain the values taken by the functions $\underline{\alpha}_2(q), \overline{\alpha}_2(q)$, for all $q \in [0, d_{2,2}]$ and by the functions $\underline{\alpha}_1(q), \overline{\alpha}_1(q)$, for all $q \in [0, d_{1,2}]$.

Let the prescribed entry point W have coordinates $(0, w)$ in the coordinate axes attached to R_1. Then we conclude that there exists a path satisfying the requirements stated in Problem 5.2 if the prescribed initial tangent angle α satisfies

$$\alpha \in [\underline{\alpha}_1(w), \overline{\alpha}_1(w)]\,.$$

In light of the recursive analysis described in Fig. 5.3, we may now concentrate on the solutions of Problems 4.4 and 4.5, as discussed in the next section.

5.2 Analysis of Traversal of a Single Rectangle

Recall from Section 4.3 that Problem 4.4 (resp. Problem 4.5) concerns finding lower and upper bounds $\underline{\alpha}$ and $\overline{\alpha}$ such that, if $\alpha \in [\underline{\alpha}, \overline{\alpha}]$, then there exists a Type 1 (resp. Type 2) admissible path Γ with $\Gamma(0) = W$ and $\Gamma'(W) = \alpha$. We adopt a constructive approach to the solution of these problems, i.e., for every point $X = (d_1, x)$ on the exit segment YZ, we construct Type 1 admissible paths Υ_x and Λ_x between the points W and X satisfying the following properties:

(P1) $\Lambda'_x(W) \leqslant \Upsilon'_x(W)$,

(P2) For every $\alpha \in [\Lambda'_x(W), \Upsilon'_x(W)]$, there exists a Type 1 admissible path from W to X with tangent angle α at W,

(P3) The tangent angles $\Upsilon'_x(W)$ and $\Lambda'_x(W)$ vary continuously with x,

(P4) There exists no other Type 1 admissible path Γ from W to X which satisfies (P1), (P2), and (P3) such that $\Gamma'(W) > \Upsilon'_x(W)$,

(P5) There exists no other Type 1 admissible path Γ from W to X which satisfies (P1), (P2), and (P3) such that $\Gamma'(W) < \Lambda'_x(W)$.

The following result enables the solution of Problems 4.4 and 4.5 using the constructions of Λ_x and Υ_x.

Theorem 5.3. *Suppose there exists a closed interval $\mathcal{I} \subset [y, z]$ such that Type 1 admissible paths (respectively, Type 2 admissible paths) Υ_x and Λ_x satisfying properties (P1)-(P5) exist for each $x \in \mathcal{I}$. Then $\overline{\alpha}$ and $\underline{\alpha}$ defined by*

$$\underline{\alpha} := \min_{x \in \mathcal{I}} \{\Lambda'_x(W)\}, \qquad \overline{\alpha} := \max_{x \in \mathcal{I}} \{\Upsilon'_x(W)\}, \tag{5.1}$$

solve Problem 4.4 (respectively, Problem 4.5).

Proof. The numbers $\underline{\alpha}$ and $\overline{\alpha}$ are well-defined by (5.1) because $\mathcal{I}$ is closed and bounded, and because $\Lambda'_x(W)$, $\Upsilon'_x(W)$ are continuous in x by (P3). We claim that for every $\alpha \in [\underline{\alpha}, \overline{\alpha}]$, there exists $x_\alpha \in \mathcal{I}$ such that $\alpha \in \left[\Lambda'_{x_\alpha}(W), \Upsilon'_{x_\alpha}(W)\right]$. To see this, note that otherwise, for any $x \in \mathcal{I}$, either $\alpha < \Lambda'_x(W) \leqslant \Upsilon'_x(W)$ or $\alpha > \Upsilon'_x(W) \geqslant \Lambda'_x(W)$. Since $\alpha \in [\underline{\alpha}, \overline{\alpha}]$, neither of these two conditions can exclusively hold for all $x \in \mathcal{I}$, and, in particular, there exists $\tilde{x} \in \mathcal{I}$ where the first condition $\alpha < \Lambda'_x(W)$ switches at $\tilde{x}$ either from true to false or vice versa. By continuity of $\Lambda'_x(W)$ in x, it follows that $\alpha = \Lambda'_{\tilde{x}}(W) \leqslant \Upsilon'_x(W)$, which contradicts the second condition $\alpha > \Upsilon'_{\tilde{x}}(W)$.

Now it follows by (P2) that there exists a Type 1 admissible path (respectively, Type 2 admissible path) Γ between W and $X = (d_1, x_\alpha)$ such that $\Gamma'(W) = \alpha$. Thus $\underline{\alpha}$ and $\overline{\alpha}$ solve Problem 4.4 (respectively, Problem 4.5). □

Computation of $\Upsilon'_x(W)$ for Problem 4.4

1: $\alpha^*(w) \leftarrow \cos^{-1}\left(1 - \dfrac{d-w}{r}\right)$

2: $n_2 \leftarrow w + \sqrt{r^2 - \left(r\sin\left(\alpha^*(w)\right) - d\right)^2} - r\cos\left(\alpha^*(w)\right)$

3: Compute $\gamma^*(x)$, for all $x \in [y, n_2]$, as the solution in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ to

$$(x-w)\cos\left(\gamma^*(x)\right) - d\sin\left(\gamma^*(x)\right) \;=\; (d^2 + (x-w)^2)/2r$$

4: **if** $\gamma^*(x) < \underline{\beta}(x)$ **then**

5: $\quad\Upsilon'_x(W)$ is the solution in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ to the equation

$$\left(\cos\underline{\beta}(x) + \frac{x-w}{r}\right)\cos\left(\Upsilon'_x(W)\right) + \left(\sin\underline{\beta}(x) - \frac{d}{r}\right)\sin\left(\Upsilon'_x(W)\right) + \frac{x-w}{r}\cos\underline{\beta}(x) - \frac{d}{r}\sin\underline{\beta}(x) + \frac{d^2 + (x-w)^2}{2r^2} \;=\; 1$$

6: **if** $\underline{\beta}(x) \leqslant \gamma^*(x) < \overline{\beta}(x)$ **then**

7: $\quad\Upsilon'_x(W)$ is the solution in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ to the equation

$$-(x-w)\cos\left(\Upsilon'_x(W)\right) + d\sin\left(\Upsilon'_x(W)\right) \;=\; (d^2 + (x-w)^2)/2r$$

8: **if** $\overline{\beta}(x) < \gamma^*(x)$ **then**

9: $\quad$There exists no Type 1 admissible path between W and X, in particular, Υ_x does not exist

Figure 5.4: Computation of $\Upsilon'_x(W)$ for the problem of traversal across parallel edges. The underlying constructions of the paths Υ_x are provided in Appendix C.

In Appendix C, we discuss the details of constructions of the paths Λ_x and Υ_x. Here we provide computational procedures based on these constructions for determining the tangent angles $\Lambda'_x(W)$ and $\Upsilon'_x(W)$. In particular, Fig. 5.4 shows the computation of $\Upsilon'_x(W)$ for the problem of traversal across parallel edges (i.e., Problem 4.4); Fig. 5.5 shows the computation of $\Upsilon'_x(W)$ for traversal across adjacent edges (i.e., Problem 4.4); and Fig. 5.6 shows the computation of $\Lambda'_x(W)$ for traversal across adjacent edges. Note that the computation of $\Lambda'_x(W)$ for traversal across parallel edges may be performed using the procedure shown in Fig. 5.4 after a reflection

Computation of $\Upsilon'_x(W)$ for Problem 4.5

1: Compute $\gamma^*(x)$, for all $x \in [y, z]$ as the solution in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ to

$$x\cos(\gamma^*(x)) - w\sin(\gamma^*(x)) \;=\; w^2 + x^2/2r$$

2: **if** $\gamma^*(x) < \underline{\beta}(x)$ **then**

3: $\Upsilon'_x(W)$ is the solution in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ to the equation

$$\left(\sin\underline{\beta}(x) - \frac{w}{r}\right)\cos(\Upsilon'_x(W)) + \left(-\cos\underline{\beta}(x) - \frac{x}{r}\right)\sin(\Upsilon'_x(W)) + \frac{x}{r}\cos\underline{\beta}(x) - \frac{w}{r}\sin\underline{\beta}(x) + \frac{w^2+x^2}{2r^2} \;=\; 1$$

4: **if** $\underline{\beta}(x) \leqslant \gamma^*(x) < \overline{\beta}(x)$ **then**

5: $\Upsilon'_x(W)$ is the solution in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ to the equation

$$w\cos(\gamma^*(x)) + x\sin(\gamma^*(x)) \;=\; w^2 + x^2/2r$$

6: **if** $\overline{\beta}(x) < \gamma^*(x)$ **then**

7: There exists no Type 2 admissible path between W and X, in particular, Υ_x does not exist

Figure 5.5: Computation of $\Upsilon'_x(W)$ for the problem of traversal across parallel edges. The underlying constructions of the paths Υ_x are provided in Appendix C.

about the horizontal, i.e., by replacing w with $d-w$, $\underline{\beta}(x)$ with $-\overline{\beta}(d-x)$, $\overline{\beta}(x)$ with $-\underline{\beta}(d-x)$, and by reversing the sign of the result.

For the sake of simplicity, the procedures in Figs. 5.4–5.6 apply to the special case where each cell in the channel is a square, and all squares of the same dimension d. Additionally, the minimum radius of turn r is assumed to satisfy $r > d$; in Appendix C, we remove both these restrictions.

Computation of $\Lambda'_x(W)$ for Problem 4.5

1: $n_1 \leftarrow r - \sqrt{r^2 - w^2}$

2: Compute $\gamma^*(x)$, for all $x \in [n_1, z]$, as the solution in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ to

$$-x\cos(\gamma^*(x)) + w\sin(\gamma^*(x)) \;=\; w^2 + x^2/2r$$

3: **if** $\overline{\beta}(x) < \gamma^*(x)$ **then**

4: $\Lambda'_x(W)$ is the solution in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ to the equation

$$\left(\sin\underline{\beta}(x) - \frac{w}{r}\right)\cos(\Lambda'_x(W)) + \left(-\cos\underline{\beta}(x) - \frac{x}{r}\right)\sin(\Lambda'_x(W)) + \frac{x}{r}\cos\underline{\beta}(x) - \frac{w}{r}\sin\underline{\beta}(x) + \frac{w^2+x^2}{2r^2} \;=\; 1$$

5: **if** $\underline{\beta}(x) \leqslant \gamma^*(x) < \overline{\beta}(x)$ **then**

6: $\Lambda'_x(W)$ is the solution in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ to the equation

$$-w\cos(\gamma^*(x)) - x\sin(\gamma^*(x)) \;=\; w^2 + x^2/2r$$

7: **if** $\gamma^*(x) < \underline{\beta}(x)$ **then**

8: There exists no Type 2 admissible path between W and X, in particular, Λ_x does not exist

Figure 5.6: Computation of $\Lambda'_x(W)$ for the problem of traversal across parallel edges. The underlying constructions of the paths Λ_x are provided in Appendix C.

Chapter 6

Multi-resolution Path- and Motion Planning

In this chapter, we consider multi-resolution planar cell decompositions such that the environment is represented with high accuracy (i.e., with small cell sizes) in the agent's immediate vicinity, and with lower accuracy in regions farther away. This approach relates to two perspectives on the practical implementations of path planning: firstly, this approach relates to the problem of appropriately *approximating large environment maps* to enable efficient online computation. To this end, the proposed multi-resolution cell decomposition significantly reduces the number of vertices in the cell decomposition graph. On the other hand, this approach also relates to the problem of *uncertainty or partial knowledge* about the environment in regions farther away from the vehicle's location. To this end, the proposed path planning scheme requires accurate environment information only locally.

In light of the prevalence of the wavelet transform in signal processing applications for autonomous navigation, as discussed in Chapter 1, we develop in the next section a multi-resolution cell decomposition scheme based on the 2-D discrete wavelet transform.

6.1 Multi-resolution Cell Decompositions using the Discrete Wavelet Transform

We define an *image* as a pair (R, F), where $R \subset \mathbb{R}^2$ is a compact, square region, and $\mathbb{L}^2(R) \ni F : R \to \mathbb{R}$ is an intensity map. We will assume that $R = \left[0, 2^D\right] \times \left[0, 2^D\right]$, with $D \in \mathbb{Z}$, and that the image intensity map F is known at a finite resolution $m_{\mathrm{f}} > -D$, i.e., the function F is piecewise constant over each of the square regions $S_{m_{\mathrm{f}},k,\ell}$, for $k, \ell = 0, 1, \ldots, 2^{D+m_{\mathrm{f}}} - 1$. In what follows, we will assume, without loss of generality, $m_{\mathrm{f}} = 0$. In the context of path planning, the intensity map F

may represent, for instance, the terrain elevation [114], a risk measure [147], or a probabilistic occupancy grid [41, 163] representing the environment.

In what follows, we will assume that the smallest cell size of interest is $2^{-m_{\mathrm{f}}} = 1$. We define a cell decomposition $\mathfrak{C}$ consisting of uniformly spaced square cells of size 1:

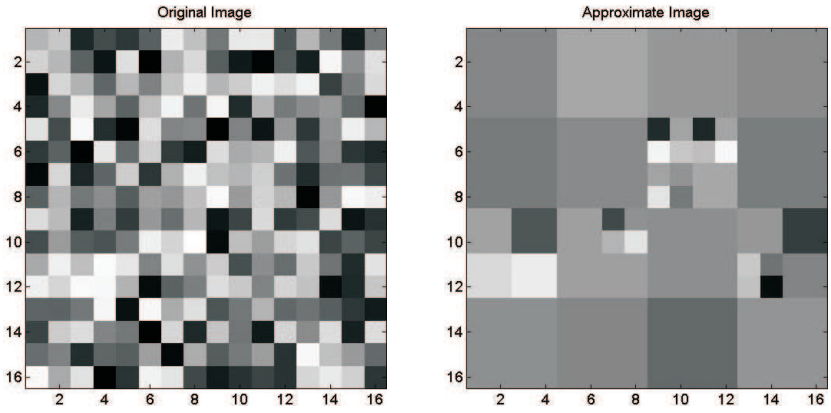
$$\mathfrak{C} := \{S_{m_{\mathrm{f}},k,\ell} : k, \ell \in \{0, 1, \ldots, 2^D - 1\}\}.$$

The intention of the geometric path planner is to find a path in the graph associated with $\mathfrak{C}$. Note that the number of cells in $\mathfrak{C}$ is 2^{2D}, which makes the graph search impractical when D is large. In the context of using multi-resolution cell decompositions to approximate large environment maps for enabling efficient online computation, the multi-resolution cell decomposition graph consists of significantly fewer vertices, thus requiring lesser computational resources for path planning at each iteration. On the other hand, in the context of availability of only partial information about the environment, the intensities of all cells in $\mathfrak{C}$ may not be known accurately due to the vehicle's sensing limitations. In this context, the multi-resolution cell decomposition scheme discussed below may be considered as a model of the vehicle's partial knowledge about the environment.

Let $a_{m_0,k,\ell}$ and $d^p_{m,k,\ell}$ be the 2-D discrete wavelet transform (DWT) coefficients of the function F, where $m_0 \in \mathbb{Z}$ is pre-specified. Let $\mathcal{A}$ be a set of triplets of integers, and let $\hat{d}^p_{m,k,\ell}$ be defined as

$$\hat{d}^p_{m,k,\ell} := \begin{cases} d^p_{m,k,\ell} & p = 1, 2, 3, \text{ and } (m, k, \ell) \in \mathcal{A}, \\ 0 & \text{otherwise.} \end{cases}$$

Then the image $(R, \hat{F})$, where $\hat{F}$ is obtained by the reconstruction (i.e., the inverse DWT) of $a_{m_0,k,\ell}$ and $\hat{d}^p_{m,k,\ell}$ is called the *approximation* of (R, F) associated with the set $\mathcal{A}$. Informally, an approximation is obtained by ignoring certain detail coefficients in the reconstruction of the wavelet transform of F; the set $\mathcal{A}$ contains the indices of detail coefficients that are "significant" for the approximation $(R, \hat{F})$.



(a) Original image (b) Approximation

Figure 6.1: Example of an image and its approximation

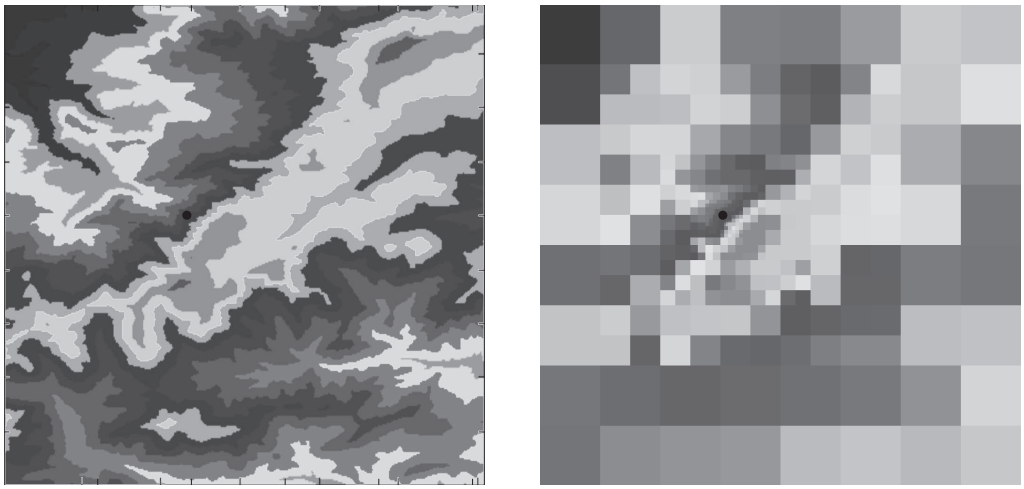
The cell decomposition $\mathfrak{C}^{\text{mr}}$ associated with $(R, \hat{F})$ is a partition of R into square cells of different sizes such that $\hat{F}$ is constant over each of the cells. For example, Fig. 6.1(a) shows the intensity map of an image of dimension 16 units, i.e. $D = 4$. Figure 6.1(b) shows the intensity map of its approximation associated with $m_0 = -2$, and the following set of significant detail coefficients:

$$\mathcal{A} \;=\; \{(-2,0,2),(-2,3,2),(-1,3,4),(-1,4,2),(-1,4,3),(-1,5,2),(-1,6,5)\}. \quad (6.1)$$

In Section 6.1.1, we describe a procedure to determine the locations, the sizes, and the values of $\hat{F}$ over each of the cells in $\mathfrak{C}^{\text{mr}}$. First, we discuss a specific approximation that is of interest in this thesis, namely, an approximation that retains detail coefficients only in the immediate vicinity of the vehicle's current position and gradually discards them in regions farther away.

Let $(x_0, y_0) \in R$ be the location of the vehicle, and let $\varrho : \mathbb{Z} \to \mathbb{N}$ be a "window" function that specifies, for each level of resolution, the distance from the vehicle's location up to which the detail coefficients at that level are significant. The set $\mathcal{A}$ of significant detail coefficients is then defined by

$$\begin{aligned}\mathcal{A} \;:=\; \big\{ & d^p_{m,k,\ell} : m_0 \leqslant m < 0, \quad p = 1,2,3, \qquad (6.2)\\ & \lfloor 2^m x_0 \rfloor - \varrho(m) \leqslant k \leqslant \lfloor 2^m x_0 \rfloor + \varrho(m), \lfloor 2^m y_0 \rfloor - \varrho(m) \leqslant \ell \leqslant \lfloor 2^m y_0 \rfloor + \varrho(m) \big\}, \end{aligned}$$



(a) Original image (b) Approximation

Figure 6.2: Example of an image and its approximation using the approximation scheme in (6.2).

where $m_0 \in \mathbb{N}$ is pre-specified. An example of an image and its approximation using (6.2) is shown in Fig. 6.2. In this example, $m_0 = -10$, and $(x_0, y_0) = (390, 449)$. The window function ϱ was chosen as the constant function $\varrho(m) = 4$ for each $m_0 \leqslant m < 0$.

6.1.1 Computing Cell Locations and Intensities

In this thesis, we use the Haar family of scaling function and wavelet. The Haar scaling function satisfies the following dilation equation [121, Sec. 2.3.2]:

$$\phi(t) \quad = \quad \phi(2t) + \phi(2t-1). \tag{6.3}$$

The dilation equation (6.3) implies that the support of the function $\phi_{m,k}$ is exactly equal to the union of the supports of the functions $\phi_{m+1,k}$ and $\phi_{m+1,k-1}$. For the 2-D case, (6.3) implies that the square support of $\Phi_{m,k,\ell}$ is exactly the union of the supports of $\Phi_{m+1,k,\ell}$, $\Phi_{m+1,k-1,\ell}$, $\Phi_{m+1,k,\ell-1}$, and $\Phi_{m+1,k-1,\ell-1}$. Consequently, an intensity map F is constant over the support of $\Phi_{m,k,\ell}$ if and only if the detail coefficients of F at level m and at higher-resolution levels $m+1, m+2, \ldots$ are all zero.

The Haar scaling function and wavelet have compact support; furthermore, the

translations of the wavelet and scaling functions are orthogonal to each other. Therefore, one may associate the detail coefficients with specific regions in $\mathbb{R}^2$, such that the values of those coefficients affect the values of the intensity map only in that region. More precisely, we make the following association:

$$d^p_{m,k,\ell} \quad \leftrightarrow \quad S_{m,k,\ell} = \left[2^{-m}k, 2^{-m}(k+1)\right] \times \left[2^{-m}\ell, 2^{-m}(\ell+1)\right], \tag{6.4}$$

for each $m_0 \leqslant m < 0$, where $m_0 \in \mathbb{Z}$ is pre-specified.

Based on the preceding observations, we present the following Rules to determine the locations and the sizes of cells in the cell decomposition $\mathfrak{C}^{\mathrm{mr}}$ associated with a given set $\mathcal{A}$ of significant detail coefficients, with $m_0 \in \mathbb{Z}$ pre-specified.

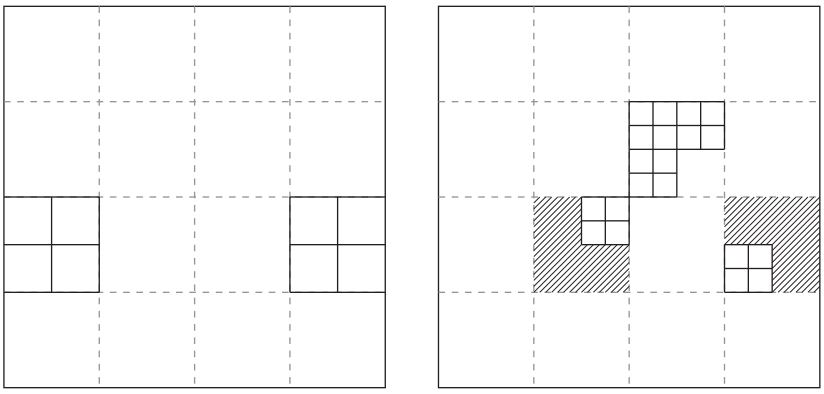
1) $\{S_{m_0,\hat{k},\hat{\ell}} : 0 \leqslant \hat{k}, \hat{\ell} < 2^{D+m_0}\} \in \mathfrak{C}^{\mathrm{mr}}$. If $\mathcal{A}$ is empty, then these cells form a uniform grid where each cell is a square of size 2^{-m_0}.

2) $\{S_{m+1,\hat{k},\hat{\ell}} : \hat{k} \in \{2k, 2k+1\},\ \hat{\ell} \in \{2\ell, 2\ell+1\}\} \in \mathfrak{C}^{\mathrm{mr}}$ whenever $(m,k,\ell) \in \mathcal{A}$. This Rule is a consequence of the fact that the support of the Haar scaling function at a given level is equal to the union of the four supports of the scaling functions at the next higher resolution level.

3) $\{S_{\hat{m}+1,\hat{k},\hat{\ell}} : \hat{k} \in \{\lfloor 2^{\hat{m}-m}k \rfloor - 1, \lfloor 2^{\hat{m}-m}k \rfloor\},\ \hat{\ell} \in \{\lfloor 2^{\hat{m}-m}\ell \rfloor - 1, \lfloor 2^{\hat{m}-m}\ell \rfloor\}, m_0 \leqslant \hat{m} < m\} \in \mathfrak{C}^{\mathrm{mr}}$, whenever $(m,k,\ell) \in \mathcal{A}$. This Rule decomposes into squares non-convex regions that arise when a detail coefficient at the level m is in $\mathcal{A}$, but detail coefficients at all levels lower than m associated with the same region (where the region of association is given by (6.4)) are not in $\mathcal{A}$.

4) $\{S_{\hat{m},\hat{k},\hat{\ell}} : \hat{k} = \lfloor 2^{\hat{m}-m}k \rfloor,\ \hat{\ell} = \lfloor 2^{\hat{m}-m}\ell \rfloor,\ m_0 \leqslant \hat{m} \leqslant m\} \notin \mathfrak{C}^{\mathrm{mr}}$, whenever $(m,k,\ell) \in \mathcal{A}$. This Rule indicates that a cell $S_{m,k,\ell}$, once decomposed, cannot belong to $\mathfrak{C}^{\mathrm{mr}}$.

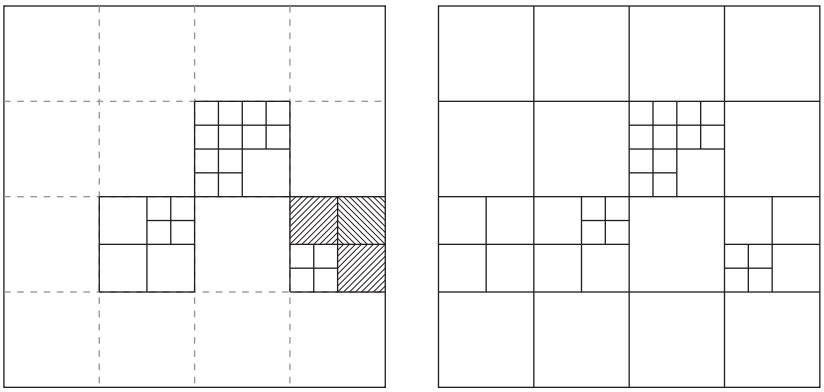
Note that some cells prescribed for inclusion in $\mathfrak{C}^{\mathrm{mr}}$ by Rule 3), are also prescribed for exclusion from $\mathfrak{C}^{\mathrm{mr}}$ by Rule 4). Furthermore, some cells are prescribed for inclusion



(a) Rule 2) applied to level 2 coefficients, along with Rule 1)

(b) Rule 2) applied to level 1 coefficients

Figure 6.3: Illustration of application of the Rules used to extract cell locations and dimensions from the wavelet coefficients.



(a) Rule 3) applied to level 1 coefficients

(b) Resultant cell decomposition

Figure 6.4: Illustration of application of the Rules used to extract cell locations and dimensions from the wavelet coefficients.

in $\mathfrak{C}^{\mathrm{mr}}$ by both Rule 2) and Rule 3). The implementation of the above Rules for determining the elements of $\mathfrak{C}^{\mathrm{mr}}$ must therefore resolve the conflict between Rules 3) and 4) and the possible redundancy. In particular, exclusions from $\mathfrak{C}^{\mathrm{mr}}$ prescribed by Rule 4) take precedence over inclusions prescribed by Rule 3); and the redundancy between Rules 2) and 3) may be avoided by checking for non-zero coefficients at lower resolutions levels associated with the same region, as given by the association (6.4).

Figures 6.3 and 6.4 illustrate the application of the preceding Rules for the approximation defined in (6.1). Figure 6.3(a) shows the grid due to the approximation

coefficients alone, and the cells due to the non-zero coefficients at the coarsest level, i.e., $m = 2$. The shaded cells in Fig. 6.3(b) illustrate the non-convex regions that may arise due to non-zero coefficients at finer levels, which need to be decomposed using Rule 3). The shaded cells in Fig. 6.4(a) are those which arise twice: due to Rule 2) for level $m = 2$ coefficients and due to Rule 3) for level $m = 3$ coefficients. Figure 6.4(b) shows the final cell decomposition, which may be compared with the actual reconstructed approximate image in Fig. 6.1(b).

Remark 6.1. After determining the elements of the cell decomposition $\mathfrak{C}^{\mathrm{mr}}(n)$, i.e., the locations and sizes of each cell in $\mathfrak{C}^{\mathrm{mr}}(n)$, the adjacency relations between cells can be determined by geometric arguments (cf. Ref. [69]). □

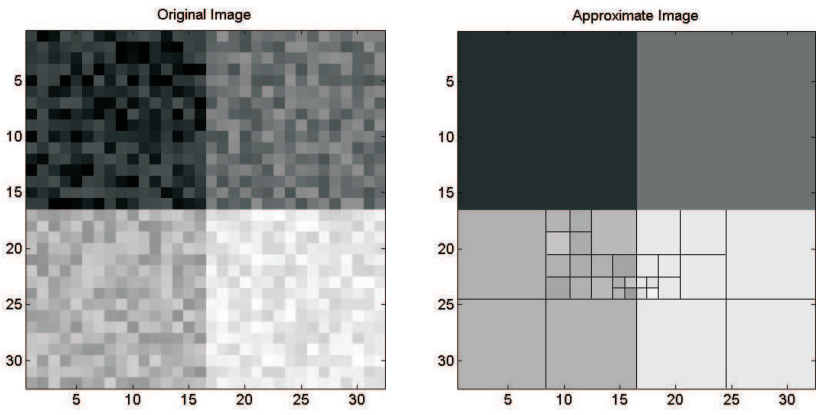
To calculate the cell intensities, we use recursively the following relations:

$$\hat{F}(S_{m_0,k,\ell}) = 2^{m_0} a_{m_0,k,\ell}, \quad 0 \leqslant k, \ell < 2^{D+m_0}, \tag{6.5}$$

$$\begin{bmatrix} \hat{F}(S_{m+1,2k,2\ell}) \\ \hat{F}(S_{m+1,2k+1,2\ell}) \\ \hat{F}(S_{m+1,2k,2\ell+1}) \\ \hat{F}(S_{m+1,2k+1,2\ell+1}) \end{bmatrix} = 2^{m_0} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 2^{-m}\hat{F}(S_{m,k,\ell}) \\ d^1_{m,k,\ell} \\ d^2_{m,k,\ell} \\ d^3_{m,k,\ell} \end{bmatrix}, \tag{6.6}$$

for $0 \leqslant k, \ell < 2^{D+m}$. In particular, (6.5) provides directly the intensities of cells arising due to Rule 1). The intensities of the cells arising by the application of Rule 2) for a triplet $(m, k, \ell) \in \mathcal{A}$ are given by (6.6). The intensities of cells arising by the application of Rule 3) for a triplet $(m, k, \ell) \in \mathcal{A}$, are each equal to $F(S_{m_1,k_1,\ell_1})$, where $(m_1, k_1, \ell_1) \in \mathcal{A}$ is the triplet with the greatest $m_1 < m$ satisfying $S_{m,k,\ell} \subset S_{m_1,k_1,\ell_1}$. If no such triplet exists, then the intensities of these cells are each equal to $F(S_{m_0,k_1,\ell_1})$, where k_1, ℓ_1 are the unique indices satisfying $S_{m,k,\ell} \subset S_{m_0,k_1,\ell_1}$.

Figure 6.5 shows another example of an image (with $D = 5$) and its approximation, with the cell decomposition superimposed for comparison. The approximation



(a) Original image (b) Approximation

Figure 6.5: Example of multi-resolution decomposition using the approximation (6.7).

corresponds to the following set of significant detail coefficients, with $m_0 = -4$:

$$\begin{aligned} \mathcal{A} \;=\; & \{(-4,0,1),(-3,1,2),(-2,2,4),(-2,2,5), \\ & (-2,3,5),(-1,7,11),(-1,8,11)\}. \end{aligned} \tag{6.7}$$

Note that, in the "fourth quadrant" region $\{(x,y) \in \mathbb{R}^2 : 16 \leqslant x < 32, \;\; 16 \leqslant y < 32\}$, only one detail coefficient, namely, $(-1,8,11)$ is significant; the cells in the fourth quadrant shown in Fig. 6.5(b) arise due to Rule 3) previously discussed.

We reiterate the fact that all information needed to define completely the cell decomposition $\mathfrak{C}^{\mathrm{mr}}$ – namely, the locations, the sizes, and the intensities of all cells – is encoded in the set $\mathcal{A}$ of significant detail coefficients, and that it is straightforward to extract this information from the set $\mathcal{A}$. Furthermore, the expression (6.2) lends itself to a fast update of the set $\mathcal{A}$ in accordance with changes in the vehicle's position in the environment. In the next section, we discuss a path planning scheme based on the multi-resolution cell decompositions discussed in this section. In the context of this path planning scheme, we also discuss the incremental computation of the set $\mathcal{A}$ (i.e., the efficient recomputation of the set $\mathcal{A}$) and the associated multi-resolution cell decomposition graph in accordance with the vehicle's motion.

6.2 Multi-resolution Path Planning

We denote by $\bar{\mathcal{G}} = (\bar{V}, \bar{E})$ the graph associated with the cell decomposition $\mathfrak{C}$, i.e., each cell in $\mathfrak{C}$ corresponds to a unique vertex in $\bar{V}$. Two vertices are adjacent if the corresponding cells are geometrically adjacent, and $\bar{E}$ is the collection of all ordered pairs $(\bar{i}, \bar{j})$ of vertices in $\bar{V}$, such that $\bar{i}$ and $\bar{j}$ are adjacent. In what follows, symbols denoting vertices, paths, or functions associated with $\mathfrak{C}$ or $\bar{\mathcal{G}}$ will be distinguished by an overline. We introduce an edge cost function $\bar{g} : \bar{E} \to \mathbb{R}_+$, which assigns to each edge of $\bar{\mathcal{G}}$ a non-negative cost of transitioning this edge.

For given initial and terminal vertices $\bar{i}_{\mathrm{S}}, \bar{i}_{\mathrm{G}} \in \bar{V}$, an *admissible path* $\bar{\pi}(\bar{i}_{\mathrm{S}}, \bar{i}_{\mathrm{G}})$ in $\bar{\mathcal{G}}$ is a finite sequence $(\bar{i}_0, \ldots, \bar{i}_{\bar{P}})$ of vertices (with no repetition) such that $\{\bar{i}_{k-1}, \bar{i}_k\} \in \bar{E}$ for each $k = 1, \ldots, P$, with $\bar{i}_0 = \bar{i}_{\mathrm{S}}$ and $\bar{i}_P = \bar{i}_{\mathrm{G}}$. For brevity, and when there is no ambiguity, we will henceforth suppress the arguments in $\bar{\pi}$. The cost $\bar{\mathcal{J}}(\bar{\pi})$ of an admissible path $\bar{\pi}$ in $\bar{\mathcal{G}}$ is the sum of costs of edges in $\bar{\pi}$, and the *path planning problem* is to find an admissible path $\bar{\pi}^*(\bar{i}_{\mathrm{S}}, \bar{i}_{\mathrm{G}})$ with minimum cost.

Next, we associate with the multi-resolution cell decomposition $\mathfrak{C}^{\mathrm{mr}}$ a graph $\mathcal{G} = (V, E)$ such that each vertex in V corresponds to a unique cell in $\mathfrak{C}^{\mathrm{mr}}$. Note that each vertex $j \in V$ also corresponds to a set $W(j, V) \subset \bar{V}$ such that the collection $\{W(j, V)\}_{j \in V}$ is a partition of $\bar{V}$. Specifically, the set $W(j, V)$ is defined by

$$W(j, V) := \left\{ \bar{j} \in \bar{V} : \mathsf{cell}(\bar{j}; \mathfrak{C}) \subseteq \mathsf{cell}(j; \mathfrak{C}^{\mathrm{mr}}) \right\}.$$

The multi-resolution cell decomposition graph $\mathcal{G}$ approximates the graph $\bar{\mathcal{G}}$ by representing each set of vertices $W(j, V) \subset \bar{V}$ with a single vertex $j \in V$. For the Haar wavelet, it can be shown that for each $j \in V$,

$$\hat{F}(\mathsf{cell}(j; \mathfrak{C}^{\mathrm{mr}})) = \frac{1}{|W(j, V)|} \sum_{\bar{j} \in W(j,V)} F(\mathsf{cell}(\bar{j}; \mathfrak{C})). \tag{6.8}$$

Finally, two vertices $i, j \in V$ are said to be adjacent in $\mathcal{G}$, i.e., $(i, j) \in E$, if and only if there exist $\bar{i} \in W(i, V)$ and $\bar{j} \in W(j, V)$ such that $\{\bar{i}, \bar{j}\} \in \bar{E}$. We will denote by

$\mathsf{cell}(j; \mathfrak{C}^{\mathrm{mr}})$ the cell in $\mathfrak{C}^{\mathrm{mr}}$ associated with the vertex $j \in V$, and by $\mathsf{vert}(S; \mathcal{G})$ the vertex in $\mathcal{G}$ associated with the cell $S \in \mathfrak{C}^{\mathrm{mr}}$.

6.2.1 Path Planning Scheme

In this section, we present a scheme to find an admissible path $\bar{\pi}$ in $\bar{\mathcal{G}}$ by repeatedly constructing multi-resolution cell decompositions and finding paths in these multi-resolution cell decompositions. This path planning scheme is a modification of the multi-resolution path planning scheme presented in [147]: these modifications ensure that the proposed path planning scheme is complete.

We assume that $F(\mathsf{cell}(\bar{j}; \mathfrak{C})) \in [0, 1]$ for each $\bar{j} \in \bar{V}$, such that more favorable cells in the environment have a lower intensity, and such that $\mathsf{cell}(\bar{j}; \mathfrak{C})$ represents an insurmountable obstacle if $F(\mathsf{cell}(\bar{j}; \mathfrak{C})) > 1 - \varepsilon$, for some pre-specified $\varepsilon \in (0, 1)$. We define the transition cost of an edge $(\bar{i}, \bar{j}) \in \bar{E}$ by

$$\bar{g}((\bar{i}, \bar{j})) := \begin{cases} \lambda_1 F(\mathsf{cell}(\bar{j}; \mathfrak{C})) + \lambda_2, & F(\mathsf{cell}(\bar{j}; \mathfrak{C})) \leqslant 1 - \varepsilon, \\ M, & \text{otherwise}, \end{cases} \tag{6.9}$$

where $\lambda_1, \lambda_2 \in (0, 1]$ and $M \gg 2|\bar{V}|$ are pre-specified constants.

We denote by $\hat{F}_n$ the approximation of the environment constructed at iteration n of the proposed algorithm, by $\mathcal{A}(n)$ the associated set of detail coefficients, by $\mathfrak{C}^{\mathrm{mr}}(n)$ the associated multi-resolution cell decomposition, and by $\mathcal{G}(n) = (V(n), E(n))$ the associated topological graph. We define the goal vertex $i_{\mathrm{G},n} \in V(n)$ as the unique vertex that satisfies $\bar{i}_{\mathrm{G}} \in W(i_{\mathrm{G},n}, V(n))$.

For each vertex $\bar{j} \in \bar{V}$, the proposed algorithm maintains an estimate $\mathcal{K}_{\mathrm{G}}(\bar{j})$ of the least cost of any path in $\bar{\mathcal{G}}$ from the vertex $\bar{j}$ to the goal vertex $\bar{i}_{\mathrm{G}}$, and a record $\mathcal{K}_{\mathrm{S}}(\bar{j})$ of the least cost of any path in $\bar{\mathcal{G}}$ from the initial vertex $\bar{i}_{\mathrm{S}}$ to the vertex $\bar{j}$. The algorithm also associates with each vertex $\bar{j} \in \bar{V}$ another vertex $b(\bar{j}) \in \bar{V}$ called the *backpointer* of $\bar{j}$. At each iteration, the algorithm performs a computation (specifically, in Line 18 or Line 20 of procedure MAIN in Fig. 6.6) whose result is a

unique vertex in $\bar{V}$. We refer to this computation as a *visit* to this vertex, and we denote by $\bar{j}_n$ the vertex visited by the algorithm at iteration n, with $\bar{j}_0 := \bar{i}_{\mathrm{S}}$. Let $j_n := \mathsf{vert}(\mathsf{cell}(\bar{j}_n; \mathfrak{C}^{\mathrm{mr}}(n)); \mathcal{G}(n))$, i.e., j_n is the vertex in $V(n)$ corresponding to the cell represented by the vertex $\bar{j}_n \in \bar{V}$.

An admissible path $\pi_n(j_n, i_{\mathrm{G},n})$ is a finite sequence $(i_0, \ldots, i_{P(n)})$ of vertices (with no repetition) in $V(n)$ that does not include the backpointer of $\bar{j}_n$ and does not include vertices in $V(n)$ corresponding to cells in the path from $\bar{i}_{\mathrm{S}}$ to $\bar{j}_n$, i.e.,

$$i_p \neq \mathsf{vert}(\mathsf{cell}(b(\bar{j}_n); \mathfrak{C}^{\mathrm{mr}}(n)); \mathcal{G}(n)), \qquad i_p \neq j_q, \quad p \in \{0, \ldots, P(n)\},$$

where $j_q \in V(n)$ is the unique vertex satisfying $\bar{j}_q \in W(j_q, V(n))$, for each $q = 0, \ldots, n-1$. Note that this definition precludes cycles in the path in $\mathcal{G}(n)$ obtained by the concatenation of the path $(j_0, \ldots, j_{n-1})$ with π_n. We introduce an edge cost function $g_n : E(n) \to \mathbb{R}_+$, which assigns to each edge of $\mathcal{G}(n)$ a non-negative cost of transitioning this edge, defined by

$$g_n((i,j)) := \begin{cases} \left(\lambda_1 \hat{F}(\mathsf{cell}(j; \mathfrak{C}^{\mathrm{mr}}(n))) + \lambda_2\right) |W(j, V(n))|, & \hat{F}(\mathsf{cell}(j; \mathfrak{C}^{\mathrm{mr}}(n))) \leqslant 1 - \varepsilon, \\ M, & \text{otherwise.} \end{cases} \tag{6.10}$$

The cost $\mathcal{J}(\pi_n)$ of the path $\pi_n(j_n, i_{\mathrm{G},n})$ is the sum of the costs of edges in the path. Note that, by (6.8) and (6.10), the cost of an obstacle-free path in $\mathcal{G}(n)$ is less than or equal to $2|\bar{V}|$, and hence a path π_n in $\mathcal{G}(n)$ is obstacle-free if and only if $\mathcal{J}(\pi_n) < M$.

The proposed path planning algorithm associates with each vertex $\bar{j} \in \bar{V}$ a binary value $\textsc{Visited}(\bar{j})$ that records whether the vertex $\bar{j}$ has previously been visited by the algorithm. That is, at any iteration of the algorithm's execution and for any $\bar{j} \in \bar{V}$, $\textsc{Visited}(\bar{j}) = 0$ indicates that the algorithm has never visited $\bar{j}$ in any previous iteration, whereas $\textsc{Visited}(\bar{j}) = 1$ indicates that the algorithm has visited $\bar{j}$ at least once during a previous iteration. The algorithm also maintains a cumulative cost $\bar{\mathcal{J}}(\bar{\pi})$ of the path $\bar{\pi}(\bar{i}_{\mathrm{S}}, \bar{j}_n)$ in $\bar{\mathcal{G}}$. The proposed multi-resolution path planning algorithm is described by the pseudo-code in Fig. 6.6. Here $x(\bar{j})$ and $y(\bar{j})$ denote, respectively, the x and y coordinates of the center of the cell $\mathsf{cell}(\bar{j}; \bar{\mathcal{G}})$.

Multi-resolution Path Planning Scheme

procedure MR-APPROX($\bar{j}$)

$\mathcal{A} \leftarrow \Big\{ d^p_{m,k,\ell} : m_0 \leqslant m < 0, \;\; p = 1,2,3,$
$\lfloor 2^m x(\bar{j}) \rfloor - \varrho(m) \leqslant k \leqslant \lfloor 2^m x(\bar{j}) \rfloor + \varrho(m),$
$\lfloor 2^m y(\bar{j}) \rfloor - \varrho(m) \leqslant \ell \leqslant \lfloor 2^m y(\bar{j}) \rfloor + \varrho(m) \Big\}.$

procedure MAIN

$\bar{\pi} \leftarrow \bar{i}_{\mathrm{S}}$, $\bar{j}_0 \leftarrow \bar{i}_{\mathrm{S}}$, $n \leftarrow 0$, $\mathsf{reachedGoal} \leftarrow 0$, $\bar{\mathcal{J}}(\bar{\pi}) \leftarrow 0$
For each $\bar{j} \in \bar{V}$, VISITED($\bar{j}$) $\leftarrow 0$
while !$\mathsf{reachedGoal}$ and $\bar{\mathcal{J}}(\bar{\pi}) < M$ and $\mathcal{K}_{\mathrm{G}}(\bar{j}_n) < M$ **do**
  $b(\bar{j}_n) \leftarrow \bar{j}_{n-1}$
  $\mathcal{A}(n) \leftarrow$ MR-APPROX($\bar{j}_n$),
  $\mathcal{G}(n) \leftarrow$ MR-GRAPH($\mathcal{A}(n)$)
  if VISITED($\bar{j}_n$) $= 1$ **then**
    $\pi^*_n = (i_0, \ldots, i_P(n)) \leftarrow \arg\min \{\mathcal{J}(\pi) : \pi$ obstacle-free in $\mathcal{G}(n)\}$, subject to $\mathcal{J}(\pi^*_n) > \mathcal{K}_{\mathrm{G}}(\bar{j}_n)$
    $\bar{\mathcal{J}}(\bar{\pi}) \leftarrow \mathcal{K}_{\mathrm{S}}(\bar{j}_n)$
  else
    $\pi^*_n \leftarrow \arg\min \{\mathcal{J}(\pi) : \pi$ obstacle-free in $\mathcal{G}(n)\}$
    $\mathcal{K}_{\mathrm{S}}(\bar{j}_n) \leftarrow \bar{\mathcal{J}}(\bar{\pi})$
  VISITED($\bar{j}_n$) $\leftarrow 1$
  if π^*_n does not exist **then**
    if $\bar{j}_n = \bar{i}_{\mathrm{S}}$ **then**
      Report failure
    else
      $\bar{j}_{n+1} \leftarrow b(\bar{j}_n)$
  else
    $\bar{j}_{n+1} \leftarrow \mathsf{vert}(\mathsf{cell}(i_1; \mathcal{G}(n)); \bar{\mathcal{G}})$
  $\mathcal{K}_{\mathrm{G}}(\bar{j}_n) \leftarrow \mathcal{J}(\pi^*_n)$
  $\mathsf{reachedGoal} \leftarrow (\mathcal{K}_{\mathrm{G}}(\bar{j}_n) = 0)$,
  $\bar{\pi} \leftarrow (\bar{\pi}, \bar{j}_n)$
  $\bar{\mathcal{J}}(\bar{\pi}) \leftarrow \bar{\mathcal{J}}(\bar{\pi}) + \bar{g}(\bar{j}_n, \bar{j}_{n+1})$
  $n \leftarrow n + 1$
if $\bar{\mathcal{J}}(\bar{\pi}) \geqslant M$ or $\mathcal{K}_{\mathrm{G}}(\bar{j}_n) \geqslant M$ **then**
  Report failure

Figure 6.6: Pseudo-code for the proposed multi-resolution path planning algorithm.

Before proving the completeness of the proposed algorithm, we make some comments regarding its execution.

Remark 6.2 (*Provision of back-tracking)***.** The preceding algorithm differs from the path planning algorithm of [147] in that we explicitly allow the algorithm to "back-track", i.e., re-visit previously visited vertices, whereas in [147], visited vertices are removed from the set $V(n)$ before finding a path in $\mathcal{G}(n)$. □

Remark 6.3. The constrained optimization problem in Line 8 can be solved by an algorithm that finds the k shortest paths in a graph. Such algorithms have been reported, for instance, in [58]. We implicitly assume that the k shortest paths will be of strictly increasing costs. This assumption is not required for the algorithm's successful execution, but it enables a concise statement of the algorithm. □

Remark 6.4 (*Cost of back-tracking)***.** Due to Line 9, the cost of "back-tracking" is not added to the cumulative cost $\bar{\mathcal{J}}(\bar{\pi})$. Also, it follows from (6.10) and Line 21 that $\mathcal{K}_{\mathrm{G}}(\bar{j}) = 0$ if and only if $\bar{j} = \bar{i}_{\mathrm{G}}$. □

Remark 6.5 (*Choice of window size)***.** The procedure MR-Approx in Fig. 6.6 and the procedure Mod-MR-Approx discussed in Section 6.2.3 assume that the window function ϱ is time-invariant, i.e., it does not change over the duration of the execution of the entire algorithm. However, this restriction is easy to remove, and it does not affect the completeness of the proposed algorithm. The window function may be required to change values to model, perhaps, a less accurate or a more accurate map of the environment, or to allow more space for recovery from cul-de-sacs at higher vehicle speeds and/or large minimum turning radii.

Notice that the algorithm in Fig. 6.6 execution at each iteration the procedure MR-Approx, and also notice that the window function ϱ is "internal" to the procedure MR-Approx. Consequently, changes of the values taken by the window function can be easily incorporated in the procedure MR-Approx without affecting

the rest of the algorithm. Moreover, the proof of completeness does make any assumptions concerning the window function, other than the assumption that the cells immediately neighboring the vehicle's current position are accurately known.

However, the *performance and optimality* of the overall algorithm does depend on the window function. In Section 7.3, we provide sample results regarding the effect of window size on the number of vertices in the multi-resolution cell decomposition graph (which is related to the performance of the algorithm), and on the optimality of the overall algorithm. Results regarding the effect of the window size on the time required for construction of the graph $\mathcal{G}(n)$ are available in [69].

The incremental computation of $\mathcal{A}(n)$, $\mathcal{G}(n)$, and $\mathfrak{C}^{\mathrm{mr}}(n)$ discussed in Section 6.2.3 specifically assume that the window function is the same as the previous iteration. Consequently, if the window function ϱ changes at iteration n, then the set $\mathcal{A}(n)$ and the graph $\mathcal{G}(n)$ must be computed "from scratch" using, respectively, procedures MR-Approx and MR-Graph. Thus, frequently changing the window function may be computationally less efficient than an invariant window function. □

6.2.2 Proof of Completeness

We associate with each path $\pi_n(j_n, i_{\mathrm{G},n}) = \{i_0, \ldots, i_{P(n)}\}$ in $\mathcal{G}(n)$ the set $\mathcal{W}(\pi_n)$ defined by

$$\mathcal{W}(\pi_n) \;:=\; \bigcup_{p=0}^{P(n)} W(i_p, V(n)). \tag{6.11}$$

The algorithm is said to *meet a setback* at iteration n if there exists no obstacle-free path $\pi_n(j_n, i_{\mathrm{G},n})$ in $\mathcal{G}(n)$ satisfying $\mathcal{W}(\pi_n) \subseteq \mathcal{W}(\pi^*_{n-1})$. A series of technical results concerning the execution of the algorithm are provided in Appendix B.2. We use these results here to prove the main result of this section.

Proposition 6.6. *The proposed algorithm is complete: if there exists an obstacle-free path in $\bar{\mathcal{G}}$ from $\bar{i}_{\mathrm{S}}$ to $\bar{i}_{\mathrm{G}}$, then the algorithm finds an obstacle-free path in a finite*

number of iterations. Otherwise, the algorithm reports failure after a finite number of iterations.

Proof. Note that because the set of vertices in $\bar{V}$ is finite, it follows by Proposition B.4 that the algorithm terminates after a finite number $N \in \mathbb{N}$ of iterations. To show completeness, first suppose that there exists an obstacle-free path in $\bar{\mathcal{G}}$ from $\bar{i}_{\mathrm{S}}$ to $\bar{i}_{\mathrm{G}}$.

Suppose that the algorithm never visits any vertex in $\bar{V}$ more than once, and that the algorithm does not meet any setbacks. By Proposition B.3, $\mathcal{K}_{\mathrm{G}}(\bar{j}_{n-1}) - \mathcal{K}_{\mathrm{G}}(\bar{j}_n) \geqslant \lambda_2$ and the sequence $\mathcal{K}_{\mathrm{G}}(\bar{j}_n)$ decreases strictly monotonically. Since $\mathcal{K}_{\mathrm{G}}(\bar{j}_n) \geqslant 0$ for each $n \in \mathbb{N}$, and since $\mathcal{K}_{\mathrm{G}}(\bar{j}_1)$ is finite (by Corollary B.2), there exists $Q \leqslant N$, such that $\mathcal{K}_{\mathrm{G}}(\bar{j}_n) = 0$ for each $n \geqslant Q$. It follows by Line 22 that the algorithm terminates after Q iterations, and since $\mathcal{K}_{\mathrm{G}}(\bar{j}_Q) = 0$, the algorithm visits the goal vertex $\bar{i}_{\mathrm{G}}$ at iteration Q.

Next, suppose that the algorithm visits some vertices in $\bar{V}$ multiple times and that the algorithm never meets any setbacks. Note that the number of multiply visited vertices is finite because the algorithm terminates in a finite number of iterations. Then either of the following statements hold: (a) the algorithm terminates at iteration $Q \leqslant N$ such that $\bar{j}_Q$ is a multiply visited vertex, or (b) there exists $Q < N$ such that for each $n = Q+1, Q+2, \ldots$, the vertex $\bar{j}_n$ is visited exactly once by the algorithm. If Statement (a) holds, then $\bar{j}_Q \neq \bar{i}_{\mathrm{G}}$ due to Lines 3 and 22, which in turn implies that the algorithm reports failure in Line 16. It follows by Line 15 that $\bar{j}_Q = \bar{i}_{\mathrm{S}}$. Then, by Proposition B.1 and Proposition B.5, there exists no admissible path in $\bar{\mathcal{G}}$ from $\bar{i}_{\mathrm{S}}$ to $\bar{i}_{\mathrm{G}}$, which is a contradiction. On the other hand, if Statement (b) holds, then by the monotonicity arguments in the preceding paragraph, the algorithm visits the goal in a finite number of iterations after iteration Q.

Next, suppose that the algorithm never visits any vertex in $\bar{V}$ more than once, and suppose that the algorithm meets some setbacks. The number of setbacks met by the algorithm is finite because the algorithm terminates in a finite number of iterations.

Then either of the following statements hold: (c) the algorithm terminates at iteration $Q \leqslant N$ such that the algorithm meets a setback at iteration Q or (d) there exists $Q < N$ such that for each $n = Q + 1, Q + 2, \ldots$, such that the algorithm does not meet any setbacks after iteration Q. Statement (c) leads to the same contradiction that follows Statement (a), whereas Statement (d) leads to the same conclusion that follows Statement (b).

Next, suppose that the algorithm visits some vertices in $\bar{V}$ multiple times and that algorithm meets some setbacks. We may combine the arguments in the two preceding paragraphs to conclude that either the algorithm visits the goal after a finite number of iterations, or (by contradiction) that there exists no obstacle-free path in $\bar{\mathcal{G}}$ from $\bar{i}_{\mathrm{S}}$ to $\bar{i}_{\mathrm{G}}$.

Finally, suppose that there exists no obstacle-free path in the graph $\bar{\mathcal{G}}$ from the initial vertex $\bar{i}_{\mathrm{S}}$ to the goal vertex $\bar{i}_{\mathrm{G}}$. The set of vertices $\bar{V}$ is finite, hence it follows by Proposition B.4 that the algorithm terminates after a finite number of iterations. Suppose, for the sake of contradiction, that the algorithm erroneously finds a path $\bar{\pi}$ from the initial vertex $\bar{i}_{\mathrm{S}}$ to the goal vertex $\bar{i}_{\mathrm{G}}$. Then $\bar{\mathcal{J}}(\bar{\pi}) > M$, since $\bar{\pi}$ is not obstacle-free. It follows by Line 24 that $\bar{\mathcal{J}}(\bar{\pi}) > M$ at some intermediate iteration of the algorithm. However, by Line 3, the algorithm terminates whenever $\bar{\mathcal{J}}(\bar{\pi}) > M$, thus leading to a contradiction. Thus, the algorithm does not erroneously find a path from the vertex $\bar{i}_{\mathrm{S}}$ to the vertex $\bar{i}_{\mathrm{G}}$ if no obstacle-free path exists, and by Line 26, it reports failure in this case. □

6.2.3 Efficient Updates of $\mathcal{A}(n)$ and $\mathcal{G}(n)$

The set of significant detail coefficients $\mathcal{A}(n)$, as computed by the procedure MR-Approx, and the associated multi-resolution cell decomposition graph both depend on the vehicle's current position. Consequently, both $\mathcal{A}(n)$ and $\mathcal{G}(n)$ must be updated in Lines 5–6 at each iteration of the algorithm in Fig. 6.6. In this section, we describe

a method to obtain $\mathcal{A}(n)$ efficiently by adding and removing elements from $\mathcal{A}(n-1)$: specifically, we first compute the elements of the sets

$$\mathcal{B}_1 := \mathcal{A}(n) \cap \mathcal{A}^{\mathrm{c}}(n-1) \quad \text{and} \quad \mathcal{B}_{-1} := \mathcal{A}(n-1) \cap \mathcal{A}^{\mathrm{c}}(n),$$

and we then evaluate the set $\mathcal{A}(n)$ by $\mathcal{A}(n) = \mathcal{A}(n-1) \cup \mathcal{B}_1 \backslash \mathcal{B}_{-1}$. To this end, we observe that for each $\bar{j} \in \bar{V}$, $x(\bar{j}) = \lfloor x(\bar{j}) \rfloor + 1/2$. It follows that for every $m \leqslant 0$,

$$\lfloor 2^m x(\bar{j}_n) \rfloor \;=\; \lfloor 2^m (\lfloor x(\bar{j}) \rfloor + 1/2) \rfloor. \tag{6.12}$$

Next, we note that

$$\lfloor x(\bar{j}_n) \rfloor \;=\; \lfloor x(\bar{j}_{n-1}) \rfloor + \Delta_x, \tag{6.13}$$

where $\Delta_x = 1$ if the vehicle moves one cell to the right at iteration n, $\Delta_x = -1$ if the vehicle moves one cell to the left at iteration n, and $\Delta_x = 0$ otherwise. It follows from (6.12) and (6.13) that

$$\begin{aligned} \lfloor 2^m x(\bar{j}_n) \rfloor &= \lfloor 2^m (\lfloor x(\bar{j}_n) \rfloor + 1/2) \rfloor = \lfloor 2^m (\lfloor x(\bar{j}_{n-1}) \rfloor + \Delta_x + 1/2) \rfloor \\ &= \lfloor \lfloor 2^m x(\bar{j}_{n-1}) \rfloor + \varsigma + 2^m \Delta_x \rfloor, \end{aligned}$$

where $\varsigma := 2^m x(\bar{j}_{n-1}) - \lfloor 2^m x(\bar{j}_{n-1}) \rfloor$. We note that

$$\varsigma \;=\; 2^m x(\bar{j}_{n-1}) - \lfloor 2^m x(\bar{j}_{n-1}) \rfloor = 2^m \left(\lfloor x(\bar{j}_{n-1}) \rfloor + 1/2 \right) - \lfloor 2^m x(\bar{j}_{n-1}) \rfloor,$$

and it follows that

$$\lfloor 2^m x(\bar{j}_{n+1}) \rfloor \;=\; \lfloor \lfloor 2^m x(\bar{j}_n) \rfloor + 2^m \Delta_x + r_x^m \rfloor, \tag{6.14}$$

where $r_x^m := 2^m \left(\lfloor 2^m x(\bar{j}_n) \rfloor + 1/2 \right) - \lfloor 2^m x(\bar{j}_n) \rfloor$. Similarly, we may show that

$$\lfloor 2^m y(\bar{j}_{n+1}) \rfloor \;=\; \lfloor \lfloor 2^m y(\bar{j}_n) \rfloor + 2^m \Delta_y + r_y^m \rfloor. \tag{6.15}$$

where $r_y^m := 2^m \left(\lfloor 2^m y(\bar{j}_n) \rfloor + 1/2 \right) - \lfloor 2^m y(\bar{j}_n) \rfloor$.

The elements of the sets $\mathcal{B}_1$ and $\mathcal{B}_{-1}$ are then determined from (6.14)-(6.15) as follows. We define scalars δ_x and δ_y as

$$\delta_x := \begin{cases} -1, & 0 > 2^m\Delta_x + r_x^m, \\ 0, & 0 \leqslant 2^m\Delta_x + r_x^m < 1, \\ 1, & 1 \leqslant 2^m\Delta_x + r_x^m, \end{cases} \qquad \delta_y := \begin{cases} -1, & 0 > 2^m\Delta_x + r_y^m, \\ 0, & 0 \leqslant 2^m\Delta_x + r_y^m < 1, \\ 1, & 1 \leqslant 2^m\Delta_x + r_y^m, \end{cases} \tag{6.16}$$

and for each $p \in \{-1, 1\}$, we define the sets $\mathcal{B}_p^{m,x}$ and $\mathcal{B}_p^{m,x}$ by

$$\begin{aligned} \mathcal{B}_p^{m,x} &:= \{(m,k,\ell) : k = \lfloor 2^m x(\bar{j}_n)\rfloor + p\delta_x, \lfloor 2^m y(\bar{j}_n)\rfloor - \varrho(m) \leqslant \ell \leqslant \lfloor 2^m y(\bar{j}_n)\rfloor + \varrho(m)\}, \\ \mathcal{B}_p^{m,y} &:= \{(m,k,\ell) : \ell = \lfloor 2^m y(\bar{j}_n)\rfloor + p\delta_y, \lfloor 2^m x(\bar{j}_n)\rfloor - \varrho(m) \leqslant k \leqslant \lfloor 2^m x(\bar{j}_n)\rfloor + \varrho(m)\}. \end{aligned}$$

Then the sets $\mathcal{B}_{-1}$ and $\mathcal{B}_1$ are given by the following relation

$$\mathcal{B}_p = \bigcup_{\alpha=\{x,y\}} \bigcup_{m_0 \leqslant m < 0} \mathcal{B}_p^{m,\alpha}, \quad p \in \{-1, 1\}. \tag{6.17}$$

The advantage of computing $\mathcal{A}(n)$ using the modified procedure Mod-MR-Approx described in Fig. 6.7 instead of computing it with procedure MR-Approx is that the sets $\mathcal{B}_{-1}$ and $\mathcal{B}_1$ have significantly fewer elements than $\mathcal{A}(n)$. More precisely, let $\bar{\varrho} := \max_{m_0 \leqslant m \leqslant 0}\{\varrho(j)\}$. Then the number of elements in the set $\mathcal{A}(n)$ is $O(\bar{\varrho}^2)$, whereas the numbers of elements in the sets $\mathcal{B}_{-1}$ and $\mathcal{B}_1$ are both $O(\bar{\varrho})$.

Furthermore, the procedure Mod-MR-Graph described in Fig. 6.7 efficiently determines the elements of the cell decomposition $\mathfrak{C}^{\rm mr}(n)$ associated with $\mathcal{A}(n)$ by adding and removing elements from $\mathfrak{C}^{\rm mr}(n-1)$, i.e., we first compute the elements of the sets $\mathfrak{C}_1^{\rm mr} := \mathfrak{C}^{\rm mr}(n) \cup (\mathfrak{C}^{\rm mr}(n-1))^{\rm c}$ and $\mathfrak{C}_{-1}^{\rm mr} = \mathfrak{C}_{-1}^{\rm mr} := \mathfrak{C}^{\rm mr}(n-1) \cup (\mathfrak{C}^{\rm mr}(n))^{\rm c}$. Alternatively, the elements of $\mathfrak{C}^{\rm mr}(n)$ can be computed directly from the set $\mathcal{A}(n)$ by executing Line 2-4 of the procedure by replacing Line 2 with the following line:

for all $(m,k,\ell) \in \mathcal{A}(n)$ **do**

Recomputation of Multi-resolution Cell Decomposition

Input: $\mathfrak{C}^{\mathrm{mr}}(n-1)$, $\mathcal{A}(n-1)$, **Output:** $\mathfrak{C}^{\mathrm{mr}}(n)$, $\mathcal{A}(n)$

procedure MOD-MR-APPROX($\mathcal{A}(n-1)$)

1: Compute $\mathcal{B}_{-1}$ and $\mathcal{B}_1$ with (6.17)

2: $\mathcal{A}(n) \leftarrow \mathcal{A}(n-1) \cup \mathcal{B}_1 \backslash \mathcal{B}_{-1}$

procedure MOD-MR-GRAPH($\mathfrak{C}^{\mathrm{mr}}(n-1), \mathcal{B}_{-1}, \mathcal{B}_1$)

1: $\mathfrak{C}^{\mathrm{mr}}_{-1} \leftarrow \varnothing$, $\mathfrak{C}^{\mathrm{mr}}_{1} \leftarrow \varnothing$

2: **for all** $(m,k,\ell) \in \mathcal{B}_1$ **do**

3: $\quad \mathfrak{C}^{\mathrm{mr}}_{1} \leftarrow \mathfrak{C}^{\mathrm{mr}}_{1} \cup \{S_{m+1,\hat{k},\hat{\ell}} : 2k \leqslant \hat{k} \leqslant 2k+1,\ \ 2\ell \leqslant \hat{\ell} \leqslant 2\ell+1\}$

4: $\quad \mathfrak{C}^{\mathrm{mr}}_{-1} \leftarrow \mathfrak{C}^{\mathrm{mr}}_{-1} \cup \{S_{\hat{m},\hat{k},\hat{\ell}} : \hat{k} = \lfloor 2^{\hat{m}-m}k \rfloor,\ \ \hat{\ell} = \lfloor 2^{\hat{m}-m}\ell \rfloor,\ \ m_0 \leqslant \hat{m} \leqslant m\}$

5: **for all** $(m,k,\ell) \in \mathcal{B}_{-1}$ **do**

6: $\quad \mathfrak{C}^{\mathrm{mr}}_{-1} \leftarrow \mathfrak{C}^{\mathrm{mr}}_{-1} \cup \{S_{m+1,\hat{k},\hat{\ell}} : 2k \leqslant \hat{k} \leqslant 2k+1,\ \ 2\ell \leqslant \hat{\ell} \leqslant 2\ell+1\}$

7: $\quad \mathfrak{C}^{\mathrm{mr}}_{1} \leftarrow \mathfrak{C}^{\mathrm{mr}}_{1} \cup \{S_{m,k,\ell}\}$

8: $\mathfrak{C}^{\mathrm{mr}}(n) \leftarrow \mathfrak{C}^{\mathrm{mr}}(n-1) \cup \mathfrak{C}^{\mathrm{mr}}_{1} \backslash \mathfrak{C}^{\mathrm{mr}}_{-1}$

Figure 6.7: Pseudo-code for the procedure MOD-MR-GRAPH.

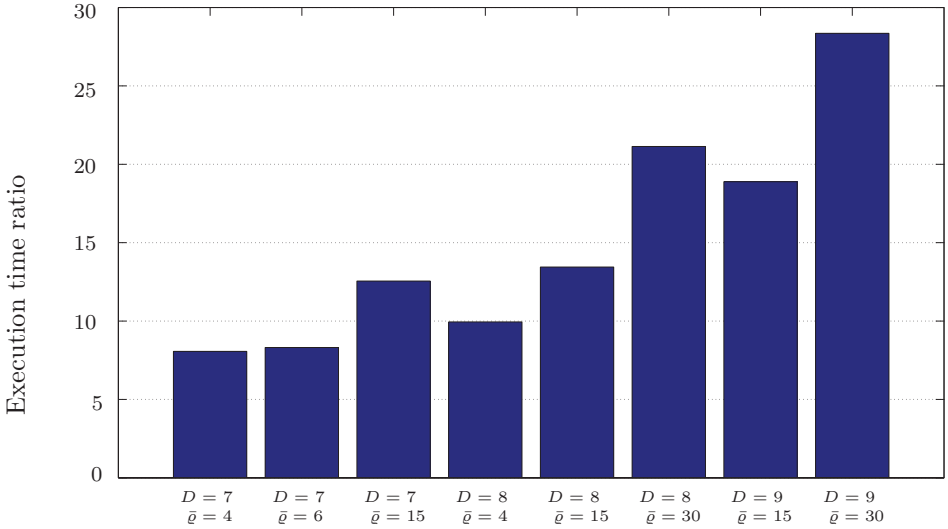
However, this approach requires $O(\bar{\varrho}^2)$ iterations of Lines 3-4 because $\mathcal{A}(n)$ has $O(\bar{\varrho}^2)$ elements. Because Lines 3-4 are constant time operations, the computation of $\mathfrak{C}^{\mathrm{mr}}(n)$ directly from $\mathcal{A}(n)$ requires $O(\bar{\varrho}^2)$ execution time. In comparison, the procedure MOD-MR-GRAPH requires $O(\bar{\varrho})$ iterations of Lines 3-4 (constant time operations) and $O(\bar{\varrho})$ of Lines 6-7 (also constant time operations), and it follows that procedure MOD-MR-GRAPH described above executes in $O(\bar{\varrho})$ time.

Remark 6.7 (*Incremental path search).* The graph $\mathcal{G}(n-1)$ can be transformed to the graph $\mathcal{G}(n)$ after adding and deleting a relatively small number of vertices and edges. In light of this observation, the operation of finding the shortest path in $\mathcal{G}(n)$ (Line 11) using standard search algorithms may seem wasteful, especially given the availability of *incremental* graph search algorithms discussed in Section A.2. However, in the context of our multi-resolution path planning algorithm, the strategy of using an incremental search algorithm to find the shortest path in the graph $\mathcal{G}(n)$ using

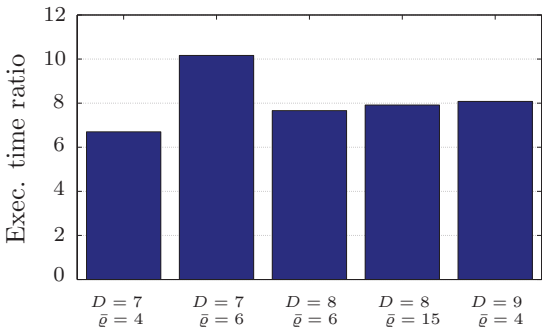
information about the shortest path in the graph $\mathcal{G}(n-1)$ does not offer significant improvements in the execution time of the overall algorithm. The primary explanation for this observation is that the number of vertices in $\mathcal{G}(n)$ is relatively small, and for graphs with a small number of vertices, incremental search algorithms do not offer significant benefits of computational efficiency [84]. □

We summarize the arguments of this section as follows: the computation of $\mathcal{A}(n)$ by the procedure MOD-MR-APPROX requires $O(\bar{\varrho})$ execution time, as compared to the procedure MR-APPROX, which executes in $O(\bar{\varrho}^2)$ time. Also, the procedure MOD-MR-GRAPH executes in $O(\bar{\varrho})$ time, as compared to direct computation of $\mathfrak{C}^{\mathrm{mr}}(n)$ from $\mathcal{A}(n)$, which executes in $O(\bar{\varrho}^2)$ time.

Figure 6.8(a) shows the results of evaluating through numerical simulations the ratio of the execution time required by the combination of the procedures MR-APPROX and MR-GRAPH to the execution time required by the combination of the procedures MOD-MR-APPROX and MOD-MR-GRAPH for computing the graph $\mathcal{G}(n)$. As predicted by the preceding theoretical analysis, the execution time ratios increase as the size $\bar{\varrho}$ of the high-resolution "window" increases. The execution time ratios also increase with the number of pixels in the environment (2^{2D}). The execution time ratios shown in Fig 6.8(a) were computed by averaging the execution time ratio over 30 simulations for each data point in Figure 6.8(a). Figure 6.8(b) shows the results of evaluating through numerical simulations the ratio of the execution time of the entire path planning algorithm using procedures MR-APPROX and MR-GRAPH to the execution time of the entire path planning algorithm using procedures MOD-MR-APPROX and MOD-MR-GRAPH. The multi-resolution path planning algorithm with the modified procedures of construction of $\mathcal{A}(n)$ and $\mathcal{G}(n)$ executes up to 10 times faster than that with the original procedures.



(a) Sample data of numerical comparisons of execution times illustrating the efficient computation of $\mathcal{A}$ and $\mathcal{G}$.



(b) Sample data of n umerical comparisons of execution times illustrating the benefits in overall path planning.

6.3 Multi-resolution Motion Planning

In this section, we discuss an extension of the preceding multi-resolution path planning scheme to include vehicle dynamical constraints. To this end, we discuss the application of H-costs in multi-resolution path planning, and we develop a multi-resolution *motion* planning scheme illustrated schematically in Fig. 1.6.

Informally, the overall motion planner searches for H-cost short paths on the graphs associated with the multi-resolution cell decompositions described in Section 6.2. However, it is unnecessary and computationally expensive to consider history-based transition costs on the entire multi-resolution cell decomposition graph

due to the following reasons: (a) large cells in $\mathfrak{C}^{\mathrm{mr}}$ correspond to coarse information about the environment in the regions associated with those cells, and hence trajectories passing through large cells will need to be refined and/or replanned in future iterations, (b) curvature-constrained paths are guaranteed to exist [14] in rectangular channels wider than a certain threshold width (compared to the upper bound on curvature), and (c) despite the relatively small numbers of vertices in the multi-resolution cell decomposition graphs, searches for H-cost short paths are computationally expensive because the numbers of vertices in the lifted graph are large.

In light of the preceding observations, and in keeping with the multi-resolution idea of using high-accuracy information only locally, the proposed motion planner searches for H-cost short paths on a "partially" lifted graph, such that the vehicle dynamical constraints are considered (via history-based transition costs) only locally. This notion of a "partially" lifted graph is stated precisely as follows.

For each $J = (j_0, \ldots, j_H) \in V_H$, and each $L \in \{1, \ldots, H-1\}$, we define the *projection* $\mathcal{P}_L(J)$ of J onto V_L, by

$$\mathcal{P}_L(J) := (j_0, \ldots, j_L) \in V_L.$$

For each $L \in \{1, \ldots, H\}$, we define the set $U_L \subseteq V_L$ by

$$\begin{aligned} U_L \;:=\; \big\{ & (j_0, \ldots, j_L) \in V_L : \mathsf{size}(\mathsf{cell}(j_k)) < \bar{d}, \\ & \text{for } k = 0, \ldots, L-1, \text{ and } \mathsf{size}(\mathsf{cell}(j_L)) \leqslant \bar{d} \big\}, \end{aligned} \tag{6.18}$$

where $\bar{d}$ is pre-specified, and $\mathsf{size}(\mathsf{cell}(j_k))$ denotes the size of the cell corresponding to the vertex j_k in the multi-resolution cell decomposition graph. By (6.18), the set U_L consists of $(L+1)$-tuples of vertices in the cell decomposition graph such that the sizes of the first L cells in each $(L+1)$-tuple are strictly lower than $\bar{d}$, whereas the size of the $(L+1)^{\text{th}}$ cell is at most $\bar{d}$. This definition alludes to the previously stated notion of including in the "partially" lifted graph only the cells small enough for the

curvature constraints to be significant. The "partially" lifted graph $\tilde{\mathcal{G}}_H = (\tilde{V}_H, \tilde{E}_H)$ is then defined by

$$\begin{aligned} \tilde{V}_H &:= U_H \cup \bigcup_{L=1}^{H-1} U_L \backslash \mathcal{P}_L(U_H), \qquad (6.19) \\ \tilde{E}_H &:= \{(I,J) : I, J \in U_H \text{ and } (I,J) \in E_H\} \cup \\ &\bigcup_{L=1}^{H-1} \left\{(I,J) : I \in U_L, J \in U_{L-1} \text{ with } [I]_1^L = J\right\}. \qquad (6.20) \end{aligned}$$

To clarify the definitions of $\tilde{V}_H$ and $\tilde{E}_H$, we provide a simple illustrative example. Consider the graph associated with the cell decomposition shown in Fig. 6.8, and let $\bar{d} = 2$ units (the various sizes of the cells shown in Fig. 6.8 are $1, 2$, and 4 units). Let $H = 2$. According to the above definitions,

$$\begin{aligned} U_1 &= \{(3,2),(3,5),\ldots,(8,7),(8,10),\ldots,(9,3),(9,6), \\ &\quad (9,8),\ldots,(10,7),(10,8)\}, \\ U_2 &= \{(3,9,8),(3,9,6),(3,9,2),\ldots,(8,9,3),(8,9,2), \\ &\quad (8,9,6),\ldots,(9,8,10),(9,8,7),\ldots,(10,3,5),\ldots\} \end{aligned}$$

Note that elements such as $(1,2),(1,5),(2,3),(4,7)$, etc. that are in V_1 do not appear in U_1. Similarly, elements such as $(3,2,1),(3,5,10)$, etc. that are in V_2 do not appear in U_2. Next, note that the projection $\mathcal{P}_1$ of U_2 onto V_1 is

$$\begin{aligned} \mathcal{P}_1(U_2) &= \{(3,9),(3,10),(8,9),(8,10),(9,3),(9,8), \\ &\quad (10,3),(10,8)\}. \end{aligned}$$

By (6.19), the set $\tilde{V}_2$ of vertices of the "partially" lifted graph is

$$\begin{aligned} \tilde{V}_2 &= U_2 \cup (U_1 \backslash \mathcal{P}_1(U_2)) \\ &= \{(3,9,8),\ldots,(10,3,5),\ldots(3,2),(3,5), \\ &\quad (8,6),(8,7),(9,2),(9,6),(10,5),(10,7)\}, \end{aligned}$$

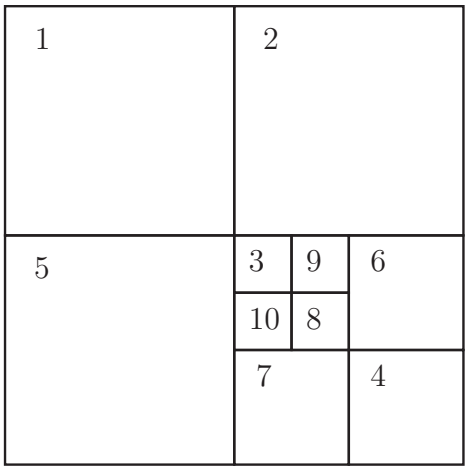


Figure 6.8: A sample multi-resolution cell decomposition.

and by (6.20), the edge set $\tilde{E}_2$ is

$$\tilde{E}_2 = \{ \overbrace{((3,9,8),(9,8,6)),((3,9,8),(9,8,7)),\ldots}^{\text{edges common with } E_2}, \underbrace{((3,9,6),(9,6)),((3,10,7),(10,7)),\ldots}_{\text{edges } (I,J) \text{ of the type } I\in U_2, J\in U_1} \}$$

The overall motion planner then operates as follows. At each iteration, a multi-resolution cell decomposition is first constructed. The cells in this decomposition may be categorized according to their sizes into two classes, namely, cells with sizes at most $\bar{d}$, and cells with sizes greater than $\bar{d}$. We define *boundary cells* in the multi-resolution cell decomposition as the cells of sizes at most $\bar{d}$ that have at least one neighboring cell in each of the two previously defined classes (see Fig. 7.13(a)). A multiple-source, single-goal implementation of the A* algorithm is used to determine the costs of optimal paths in the multi-resolution cell decomposition graph from the vertices associated with each of the boundary cells to the goal vertex. These costs are then used as terminal penalty costs in the execution of the H-cost path planner on the "partially" lifted graph previously discussed. This H-cost path planner returns a sequence of cells from the current location to one of the boundary cells, along with an admissible vehicle control input that enables the traversal of this sequence of cells. The vehicle state is advanced by traversing one cell using this control input, and the

process is repeated until the vehicle reaches the goal.

Chapter 7

Simulation Results and Discussion

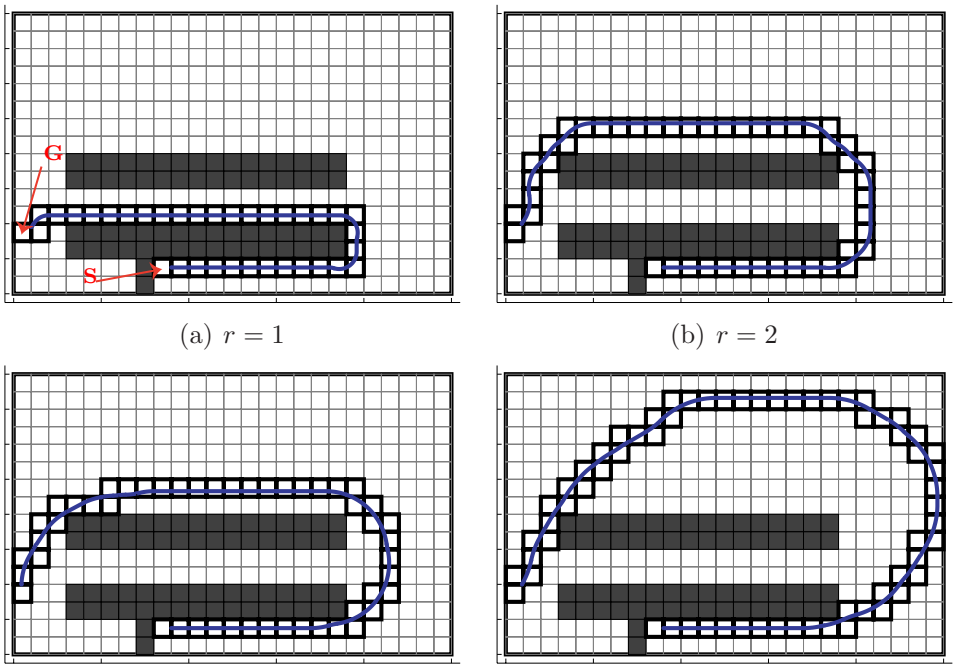
7.1 Path Planning for the Dubins Car

As discussed in Section 4.4.1, the upper bound on the curvature of feasible geometric paths for the Dubins car model is $\kappa_{\max} = (rv)^{-1}$. The configuration space $\mathcal{C}$ and the state space $\mathcal{D}$ are identical, and hence the effective target sets coincide with the effective target configurations sets. Additionally, the feasible paths specified in TilePlan can also be constructed geometrically.

Figure 7.1 shows results of the simulations of the proposed motion planner for the Dubins car model in an environment similar to the motivating example of Chapter 2. Note, in particular, that different channels are obtained for different bounds on the curvature, whereas *any* cost function defined on single-edge transitions of the cell decomposition graph $\mathcal{G}$ will result in the channel shown in Fig. 7.1(a). For this example, the H-cost of an edge in $\mathcal{G}_H$ was defined as the time of traversal of the tile corresponding to that edge, i.e., $\ell(\xi, u, t) := 1$ in (3.5).

Figure 7.2 shows the results of the simulations of the proposed motion planner for the Dubins car model in a maze-like environment. For this example, as before, the H-cost of an edge in $\mathcal{G}_H$ was defined as the time of traversal of the tile corresponding to that edge. Note the significant difference in the resultant channel when different curvature bounds (in accordance, perhaps, with different speeds of travel or different steering rate constraints) are imposed. It should be stressed that in each of the sample results shown in Figs. 7.1 and 7.2, the sequence of cells from the initial cell to the goal cell is obtained by the proposed motion planner *simultaneously* with the curvature-bounded path lying within that channel.

As previously mentioned in Section 3.1, when a uniform cell decomposition is



(a) $r = 1$ (b) $r = 2$

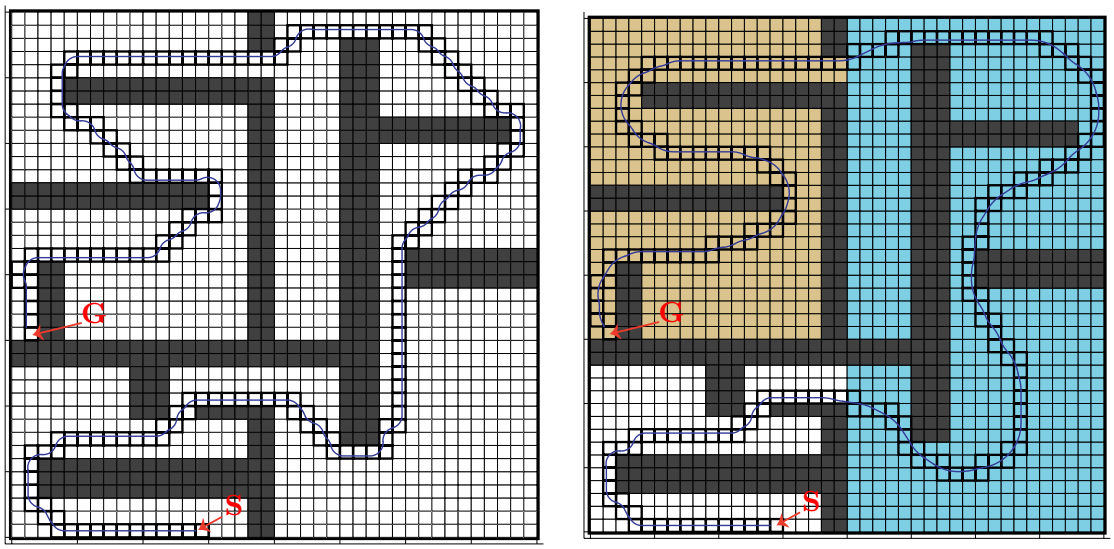
(c) $r = 3.5$ (d) $r = 4.5$

Figure 7.1: Making U-turns with different curvature bounds. The curvature bound in each case is r^{-1}. Here, $H = 3$.

used, the rectangular channel corresponding to any tile belongs to an equivalence class of geometrically equivalent (up to the rigid geometric transformations) tiles. The number of such equivalence classes is a small, finite number (e.g., there are 13 equivalence classes with $H = 3$). To reduce the execution time of the overall motion planner, we may pre-process offline the computation of effective target sets for each equivalence class, and perform only the synthesis of paths iteratively during the execution of the motion planner. The sample results shown in Figs. 7.1 and 7.2 were obtained using such pre-processed computations.

7.2 Comparative Simulation Results and Discussion

In Section 1.3, we discussed briefly and qualitatively the advantages of the proposed motion planning framework over existing approaches in the literature. Here, we discuss these advantages in detail, and we corroborate our claims with numerical simulation data.



(a) Uniform curvature bound, $r = 1$

(b) Patch-wise varying curvature bound, $r_{\text{white}} = 1$, $r_{\text{yellow}} = 2.25$, $r_{\text{blue}} = 3.5$.

Figure 7.2: Simulation result: simple maze-like environment. The curvature bound in each case is r^{-1}. For this simulation, we chose $H = 3$.

7.2.1 Comparisons with Randomized Sampling-based Motion Planners

As discussed in Section 1.2.3, randomized sampling-based algorithms based on RRTs [94] are fast algorithms for solutions of the point-to-point motion planning problem in high dimensional spaces. In this section, we present a thorough comparison of the proposed motion planner with RRT-based algorithms. In particular, we present numerical simulation data demonstrating the superiority of the proposed motion planner in terms of the costs of resultant trajectories. For the simulation results that follow, we considered the particle dynamical model discussed in Section 4.4.2, and as before, we defined the H-cost of an edge in $\mathcal{G}_H$ as the time of traversal of the tile corresponding to that edge. We implemented TilePlan using the based on MPC and effective target sets, discussed in Chapters 4 and 5.

We compared the proposed motion planner against the following two RRT-based

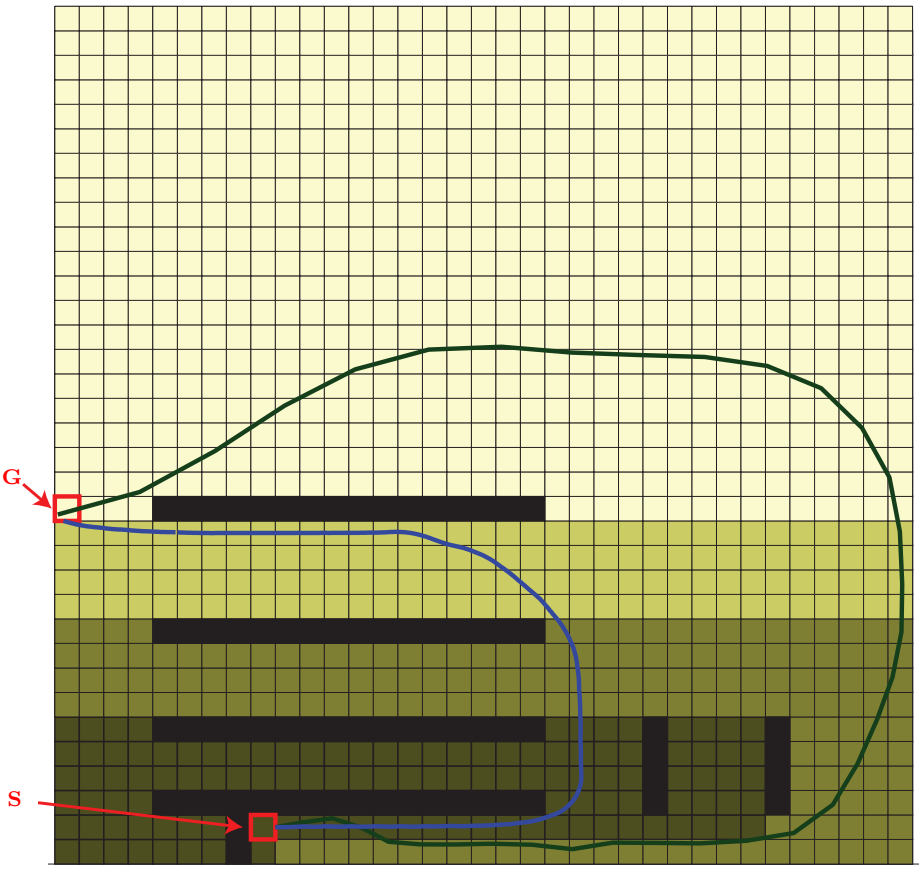
planners: (1) the standard RRT-based planner[1] as reported in [94], and (2) the T-RRT planner recently reported in [64]. The T-RRT planner finds low-cost trajectories with respect to a pre-specified state space[2] cost map. Note that the minimum-time criterion cannot be expressed as a state space cost map; therefore, we execute the T-RRT planner with the objective "travel as fast as possible," which is immediately defined by the state space cost map $c(x, y, \theta, v) = v$.

Figure 7.3(a) shows the first of two environments used for the comparative numerical simulation examples. This environment consists of "lanes" separated by obstacles (black regions), with a different upper bound on the allowable speed of the vehicle (lighter areas represent higher upper bounds). The "friction ellipse" parameters were fixed at $f_{\mathrm{r}}^{\max} = 1$, $f_{\mathrm{t}}^{\max} = 0.25$ over the entire environment. The initial and goal cells are marked in Fig. 7.3(a); as before, the objective was to find a minimum time trajectory from the initial cell to the goal cell.

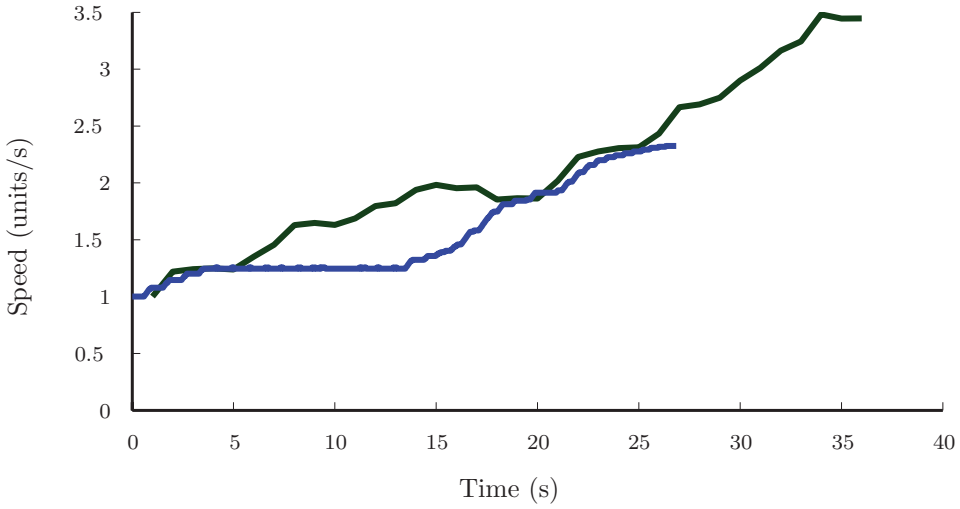
As mentioned previously, linear interpolation between two states does not, in general, correspond to a feasible state trajectory. Hence, for extending known states towards randomly selected new states, the RRT-based planners were programmed to randomly select an input vector from the set of admissible inputs and integrate the vehicle model for a fixed time δ, as recommended in [94]. For the "lanes" environment, we used three different values of δ, namely, $\delta = 0.5$s, $\delta = 1$s, and $\delta = 1.5$s, and we conducted 30 trials of both algorithms (standard RRT and T-RRT) for each value of δ. For comparison, we executed the proposed algorithm on the same environment with three different values of H, namely, $H = 4$, $H = 5$, and $H = 6$, with $L = 10$ in each case.

[1]Several improvements to the standard RRT planner of [94] have been reported (see [64,118] and references therein); however, with the exception of [64], these improvements focus mainly on the efficiency of the motion planner. We choose to compare against the standard RRT planner as we are mainly interested in the comparison of costs of resultant trajectories.

[2]In [64], the authors deal with a configuration space cost map, but their approach extends easily to state spaces.



(a) Geometric paths



(b) Speed profiles

Figure 7.3: The "lanes" environment. Black colored regions are obstacles; areas with other colors represent different speed limits: $v_{\max} = 1.25$ units/s for the darkest area, $v_{\max} = 2$ units/s, $v_{\max} = 2.5$ units/s, and $v_{\max} = 3.5$ units/s, respectively, for progressively lighter areas. The dark green curve is the geometric path corresponding to a sample trajectory returned by the T-RRT algorithm, with $\delta = 1$ s. The blue curve is the geometric path corresponding to the trajectory returned by the proposed approach, with $H = 6$.

Figure 7.5(a) shows comparative data for the trajectory costs (i.e., time of traversal) resulting from the simulations described above. The proposed motion planner returned trajectories with almost identical costs for each H, in particular, the trajectory cost corresponding to $H = 6$ was 26.626 s. On the other hand, both the standard RRT and T-RRT planners returned, on an average, significantly costlier trajectories. For instance, the trajectory costs returned by the standard RRT planner with $\delta = 1$ were, in the best case 24% higher, on an average 78% higher, and in the worst case 181% higher. Similarly, the trajectory costs returned by the standard RRT planner with $\delta = 1$ were, in the best case 8.9% higher, on an average 29% higher, and in the worst case 46% higher.

Figure 7.3(a) shows the geometric path corresponding to the trajectory returned by the proposed planner with $H = 6$ (blue curve) in comparison to the geometric path corresponding to a trajectory returned by the T-RRT planner in one of the 30 trials with $\delta = 1$ (green curve); Fig. 7.3(b) shows the speed profiles corresponding to these two trajectories. This example illustrates that the "travel as fast as possible" objective is not always a practically acceptable alternative to the minimum-time criterion: Figure 7.3(b) shows that the vehicle achieves higher speeds along the T-RRT trajectory but the travel time is 35.2% higher than the trajectory found by the proposed planner. This result is a consequence of the input constraint (4.15), which forces the vehicle to traverse paths of lower curvature at higher speeds, thus producing longer geometric paths.

Figure 7.4 shows the second, maze-like environment used for our comparative analysis. As before, different upper bounds on the speed were assigned to different areas in the environment, and the friction ellipse parameters were fixed at $f_{\mathrm{r}}^{\max} = 1$, $f_{\mathrm{t}}^{\max} = 0.25$ over the entire environment. As before, the objective is to find a minimum-time trajectory from the initial cell to the goal cell. We compared the

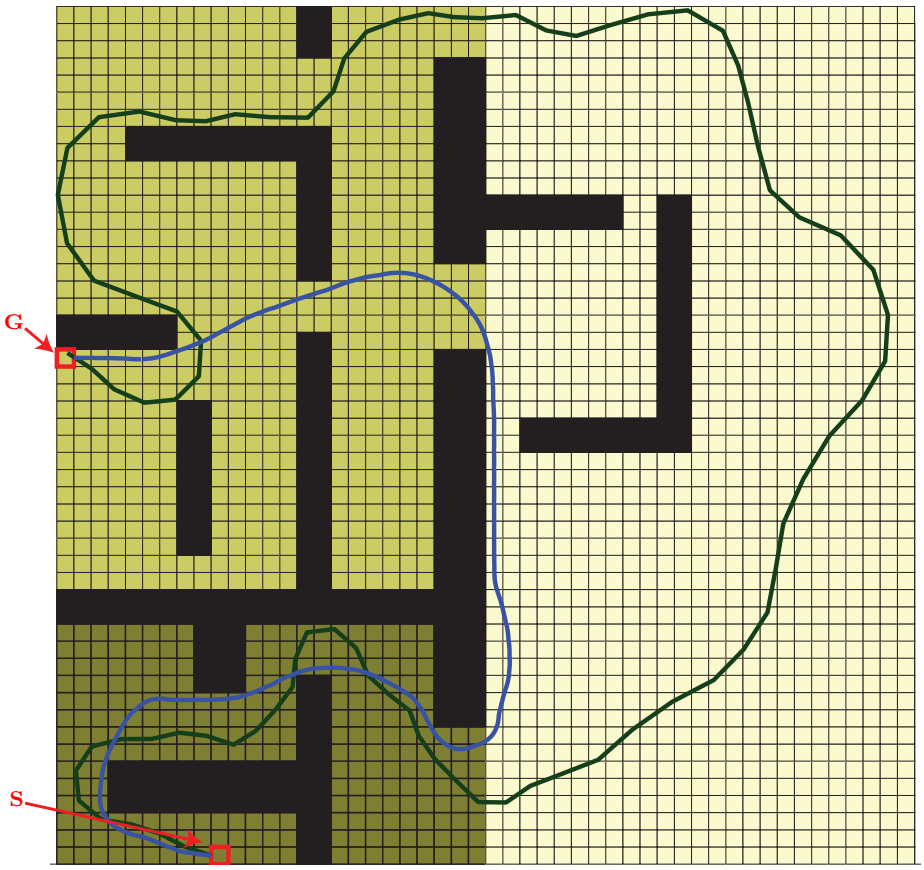
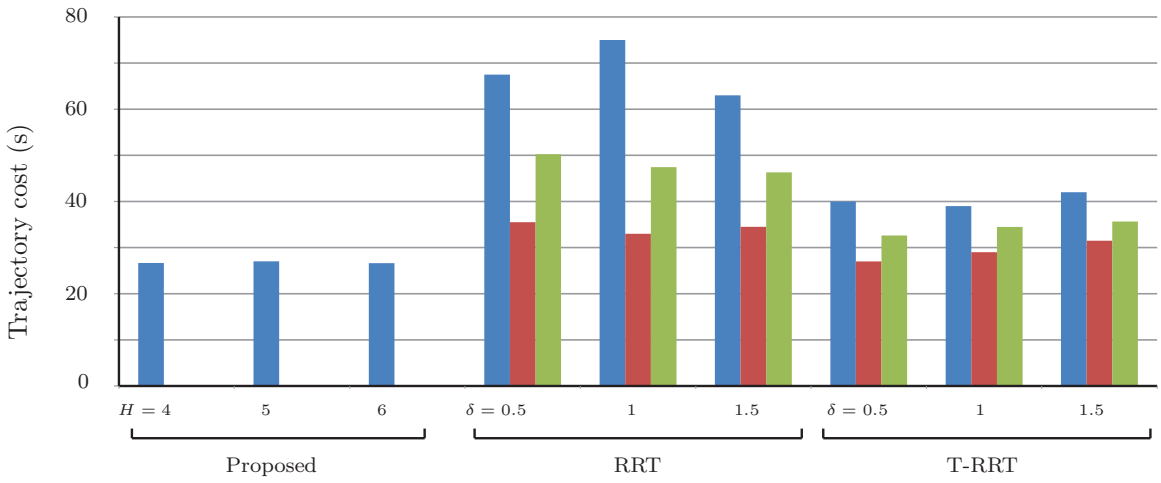


Figure 7.4: The "maze" environment. Black colored regions are obstacles; areas with other colors represent different speed limits: $v_{\max} = 1.25$ units/s for the darkest area, $v_{\max} = 2$ units/s, and $v_{\max} = 2.25$ units/s, respectively, for progressively lighter areas. The dark green curve is the geometric path corresponding to a sample trajectory returned by the RRT-based planner, with $\delta = 1.5$ s. The blue curve is the geometric path corresponding to the trajectory returned by the proposed approach, with $H = 5$.

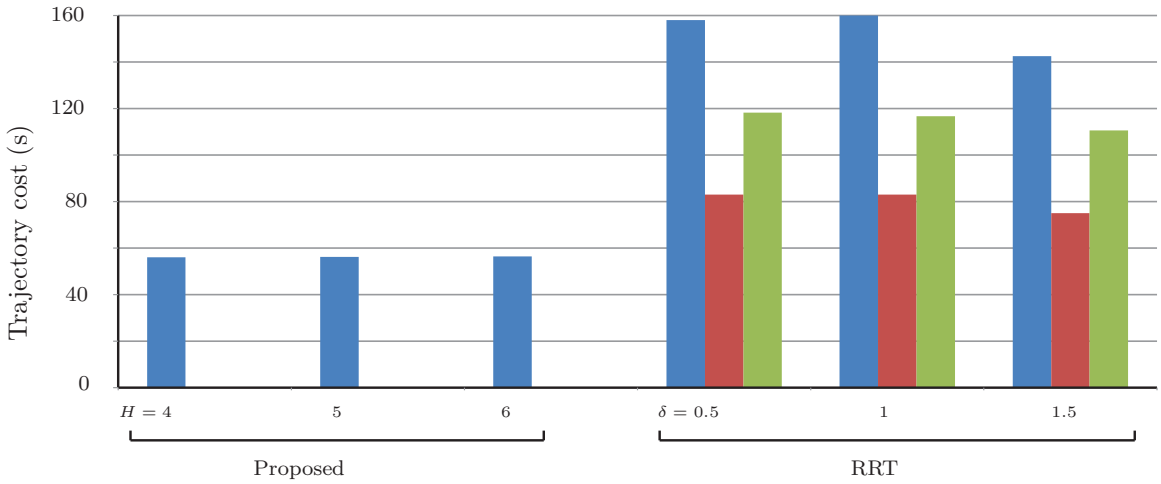
proposed motion planner against the standard RRT planner alone, because the T-RRT planner was found to be impractically slow for this case. As shown in Fig. 7.4, the environment has a narrow "short-cut" between the initial cell and the goal cell.

Figure 7.5(b) shows comparative data for the trajectory costs for this maze-like environment. The proposed motion planner returned trajectories with almost identical costs for each H; in particular, the trajectory cost corresponding to $H = 5$ was 56.23 s. The trajectory costs returned by the standard RRT planner were significantly higher, mainly because it failed to traverse the aforementioned "short-cut" on several occasions, as illustrated in Fig. 7.4. For instance, the trajectory costs returned by the standard RRT planner with $\delta = 1$ were, in the best case 48% higher, on an average 107% higher, and in the worst case 185% higher. Clearly, the average costs of trajectories returned by RRT-based planners may be further worsened in environments where the differences between the costs of trajectories corresponding to "short-cuts" and the costs of alternative trajectories is larger.

Figure 7.6(a) shows on a logarithmic scale the number of states explored by each of the algorithms discussed in the previous section for the "lanes" environment; Fig. 7.6(b) shows similar data for the maze-like environment. In both cases, the number of states explored by the RRT-based planners were higher by at least an order of magnitude. However, the time required for the RRT-based planners to explore a new state (including the nearest neighbor search and collision checking) was found to be approximately an order of magnitude lower than the execution time of the MPC-based TilePlan. Consequently, the comparison of the performance of the proposed planner against the aforementioned RRT-based planners showed no conclusive evidence of the superiority of either planner over the other in that respect. Direct comparisons of the execution times of these algorithms corroborated this observation. However, Fig. 7.6 shows that the proposed planner is preferable in cases where the exploration of new states is expensive due to complex dynamics or due to expensive collision



(a) Data for the "lanes" environment in Fig. 7.3(a).



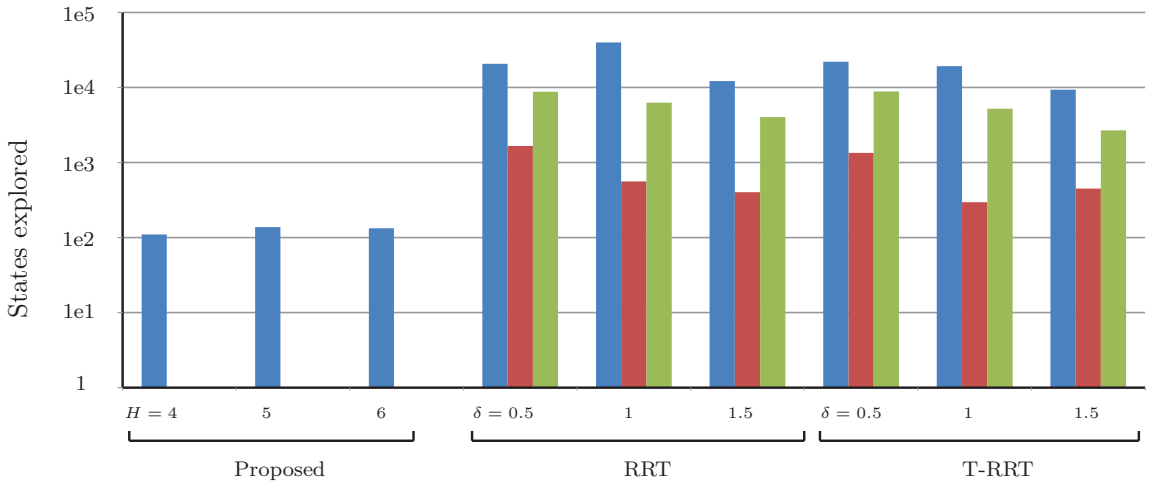
(b) Data for the maze-like environment in Fig. 7.4.

Figure 7.5: Comparison of trajectory costs: for the RRT and T-RRT data, the blue (left), red (middle), and green (right) bars represent, respectively, the maximum, the minimum, and the average values over 30 trials.

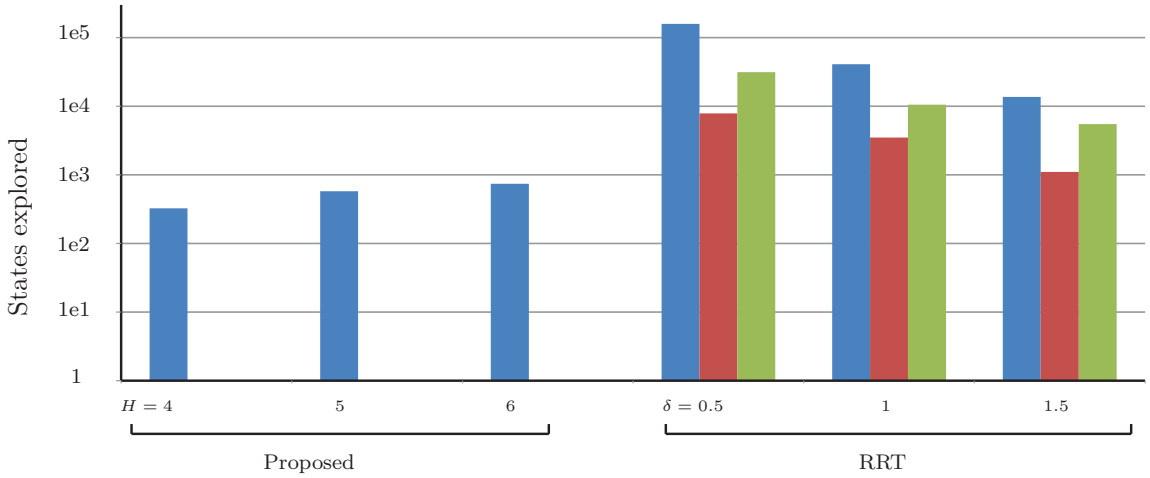
checking. In this context, it may be noted that the number of states explored in RRT-based planners can be significantly reduced by "intelligent" sampling heuristics that ensure efficient coverage of the entire search space (cf. Refs. [64, 65, 118, 162]).

7.2.1.1 Qualitative Comparisons with Randomized Sampling-based Motion Planners

The exploration of the state space is difficult with standard RRT-based motion planners when the states and control inputs are coupled via complex, nonlinear differential equations [118], because linear interpolation between two states no longer corresponds,



(a) Data for the "lanes" environment in Fig. 7.3(a).



(b) Data for the maze-like environment in Fig. 7.4.

Figure 7.6: Comparison of number of states explored: for the RRT and T-RRT data, the blue (left), red (middle), and green (right) bars represent, respectively, the maximum, the minimum, and the average values over 30 trials.

in general, to an admissible state trajectory. Whereas [118] and similar earlier works focus on aiding the *efficiency* of sampling-based algorithms using a discrete search, we focus on the complementary aspect of *optimality* by ensuring that the result of a discrete shortest-path search remains compatible with the vehicle dynamics.

In addition to the benefits of the proposed planner over randomized sampling-based planners in terms of optimality, the proposed planner also offers the benefit of a clear distinction between the discrete and continuous layers of motion planning.

The idea of planning on the lifted graph, introduced in Chapter 2, allows this distinction to be maintained, while providing guarantees of consistency between the two levels of planning. Consequently, changes to the discrete planning strategy and/or the (continuous) tile motion planning may be incorporated with relative ease. In this paper, we used the shortest-path search as a concrete, important example of a discrete search strategy; in the future, we envision extensions of the proposed planner where the discrete planner attempts to satisfy vehicular tasks specified as formulae of predicate or temporal logic [12] instead of solving a shortest path problem. On the other hand, complex vehicle dynamics can be easily incorporated in TilePlan without affecting the discrete planning strategy.

In the context of the shortest path problem alone, different trajectory quality criteria can be incorporated easily to meet different motion planning objectives. For example, Fig. 7.7 shows the result of simulating the proposed planner by defining the H-cost as a weighted sum of the time of traversal and the terrain elevation. Consequently, the planner finds paths that traverse low-elevation portions of the terrain (lighter regions in Fig. 7.7), while ensuring kinematic feasibility guarantees of the resultant paths. The Dubins car model was used for this simulation; the two curves in Fig. 7.7 indicate the resultant paths for different curvature constraints. It should be noted that the problem of finding low cost trajectories with respect to configuration space cost maps using randomized sampling-based methods has been addressed in [64]; however, many important trajectory quality criteria such as time optimality (considered in the preceding section) and fuel optimality cannot be expressed as configuration space cost maps.

7.2.2 Comparisons with Feedback-based Motion Planners

An underlying assumption in the feedback-based motion planning approach described in Section 1.2.4 is the complete controllability of the vehicle dynamical model in the

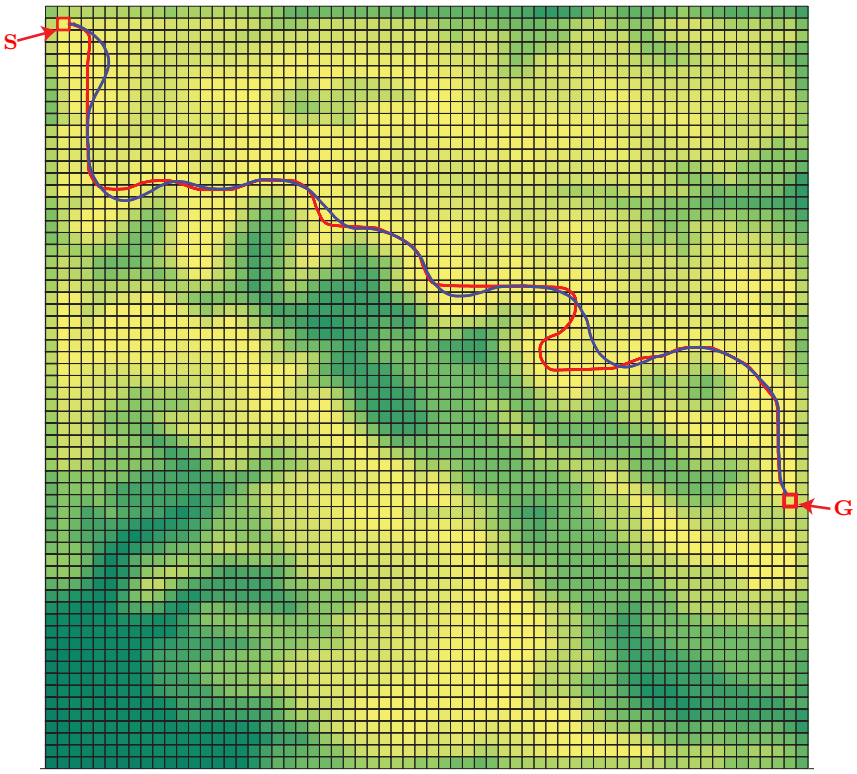


Figure 7.7: Application of proposed motion planning framework for finding "low-elevation" paths: lighter areas represent more favorable regions of the workspace. The Dubins car kinematical model is used in TilePlan. The red curve represents a path with $r = 1$, whereas the blue curve represents a path with $r = 2$.

presence of obstacles, i.e., the assumption that there exists a feasible, obstacle-free trajectory from any initial state to any goal state. In the context of mobile vehicles, complete controllability in the presence of obstacles is a strong assumption: fixed-wing aircraft moving in the horizontal plane do not satisfy this assumption; terrestrial vehicles constrained to move only forward, or high-speed vehicles for which stopping and reversing the direction of motion may not be desirable, also do not satisfy this assumption. In contrast with the planners presented in [13, 32, 99–101], the proposed motion planning framework does not assume complete controllability in the presence of obstacles.

When the complete controllability assumption is violated, the central tenet of the preceding feedback-based motion planning schemes is no longer valid: arbitrary

j	-9	-8	-7	-6	-5	-4	-3	-2	-1
ϱ_1	1	1	2	2	2	2	2	2	3
ϱ_2	1	2	2	3	4	5	6	7	7
ϱ_3	3	3	5	5	6	7	8	8	9

Table 7.1: Values of the three window functions chosen for simulating the multi-resolution path planning algorithm

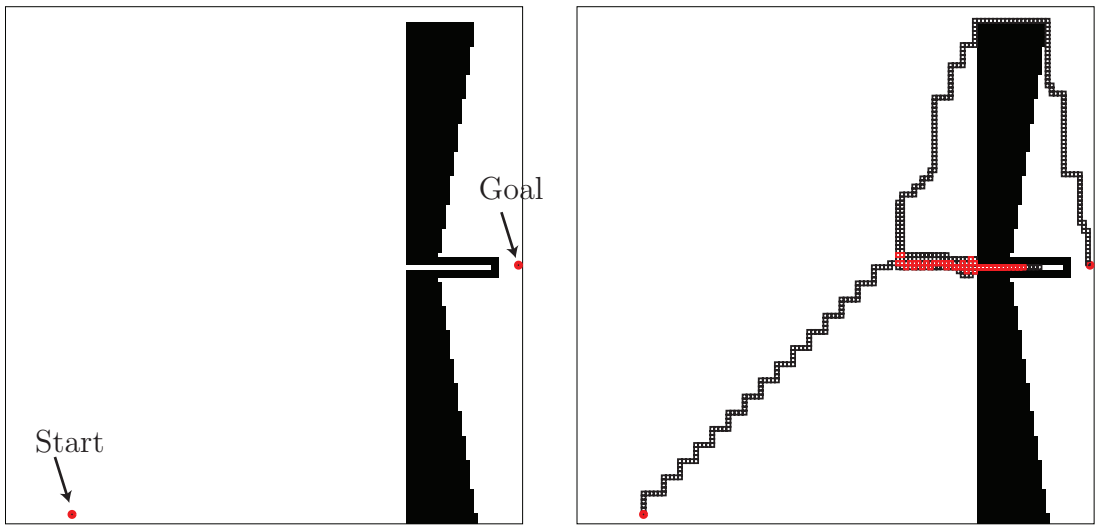
sequences of cell transitions cannot *in principle* be guaranteed from arbitrary initial states. A simple example of a vehicle kinematic model that violates the complete controllability assumption is the Dubins car model. For any given sequence of cell transitions in the workspace, there exists a set of initial states of the vehicle from which it is impossible for the vehicle to execute that sequence. The proposed framework does not require this assumption because the geometric planner ensures by computing an admissible control input, given in (3.6), the feasibility of traversal of its resultant path (i.e., the sequence of cell transitions from the initial position to the goal).

7.3 Multi-resolution Path- and Motion Planning

In this section, we present numerical simulation results of implementations of the multi-resolution path- and motion planning schemes presented in Chapter 6. First, we focus on the path planning algorithm, which does not consider vehicle dynamics.

7.3.1 Optimality and Performance of the Path Planning Scheme

Recall that in Section 6.2.2, we proved rigorously the completeness of the multi-resolution path planning algorithm, which concerns the algorithm's capability of finding a *feasible* path whenever such a path exists. Figures 7.8 and 7.9 illustrate a simulation example demonstrating the capability of the multi-resolution path planning scheme to recover from a cul-de-sac. As shown in Fig. 7.8(a), we designed the shape of the obstacle and the location of the goal to lead the multi-resolution path planning algorithm into the cul-de-sac in the "central" region of the obstacle, whereas the goal can only be reached from the "top" region of the obstacle. Figure 7.9



(a) Map of the environment.

(b) Final path found by the multi-resolution path planning algorithm: the red-colored cells were multiply visited by the algorithm.

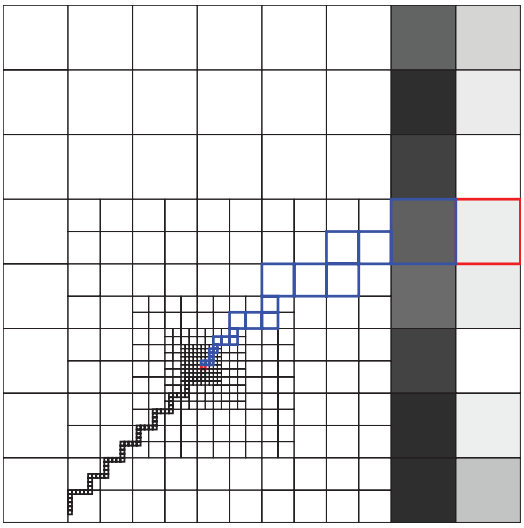
Figure 7.8: Demonstration of the multi-resolution path planning algorithm's ability to recover from a cul-de-sac.

illustrates some intermediate iterations in the execution of the multi-resolution path planning algorithm on this environment; in particular, the algorithm leads the vehicle into the cul-de-sac but in later iterations, it successfully recovers and finds a path to the goal.

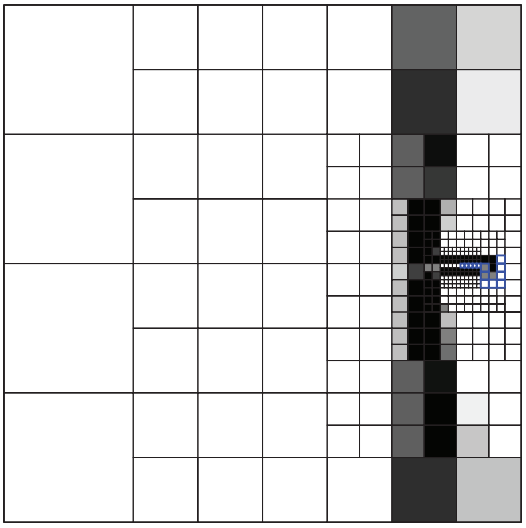
Whereas we can guarantee the algorithm's capability of finding a *feasible* path whenever such a path exists, we do not yet have theoretical results concerning the *optimality* of the resultant path. Consequently, we study using numerical simulation results the optimality of paths resulting from the multi-resolution path planning algorithm.

To this end, we compared the cost of the resultant paths with the cost of an optimal path found by executing the A* algorithm on the graph $\bar{\mathcal{G}}$ associated with the finest-level cell decomposition.

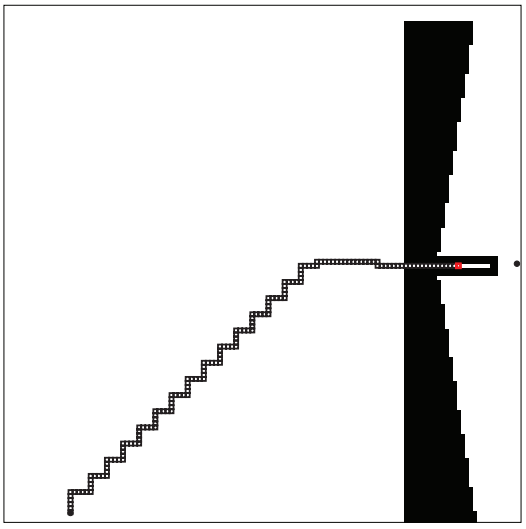
For these comparative simulations, we chose an environment represented by the image shown in Fig. 6.2(a), and we defined the cost of a path in $\bar{\mathcal{G}}$ by (6.9), with $\lambda_1 = 1$ and $\lambda_2 = 0.1$. We chose three different "window" functions, as described in



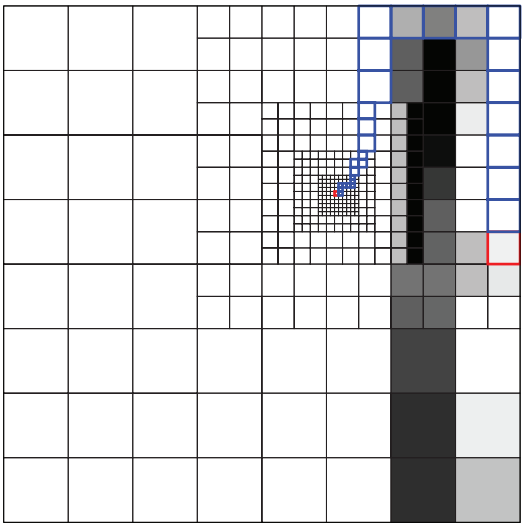
(a) An intermediate iteration before the cul-de-sac is explored. Note that due to the peculiar shape of the obstacle space, the "central" region of the obstacle containing the cul-de-sac is approximated by cells of lower intensities than the cells approximating the "top" region of the obstacle.



(b) The intermediate iteration at which the cul-de-sac is encountered, i.e., the algorithm fails to find an admissible path (as defined in Section 6.2) from the vehicle's current location to the goal.



(c) The location of the vehicle in the environment at the iteration illustrated in Fig. 7.9(b).



(d) The intermediate iteration (after encountering the cul-de-sac) at which the path planning algorithm finds a multi-resolution channel containing the actual path to the goal.

Figure 7.9: Intermediate iterations in the multi-resolution path planning algorithm's implementation for the environment shown in Fig. 7.8(a).

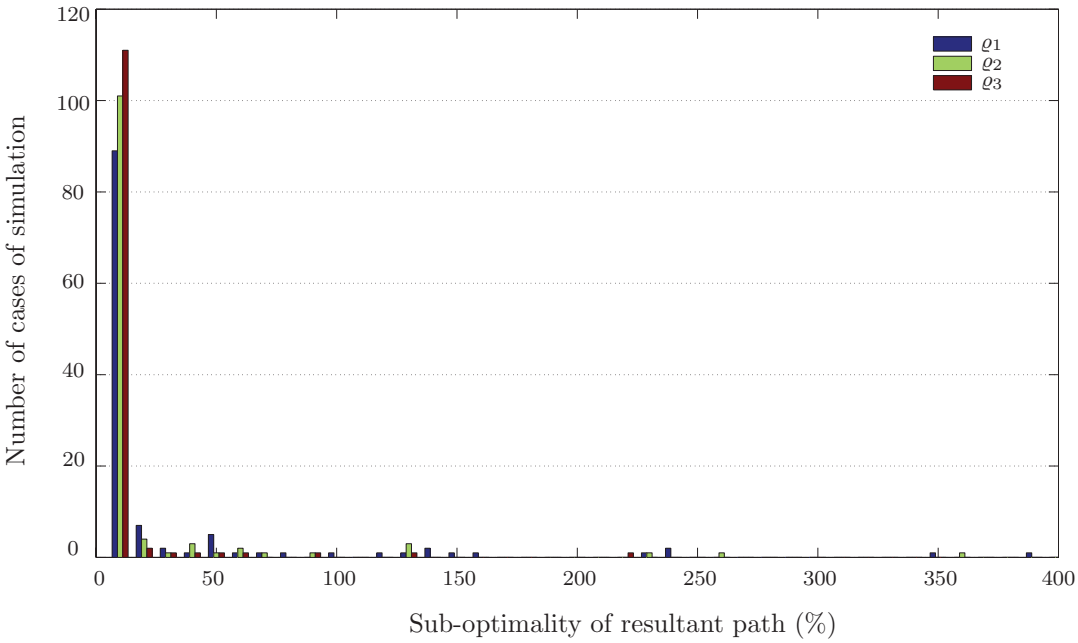


Figure 7.10: Histogram showing the distribution according to percentage sub-optimality of simulated cases, for different window functions. Whereas the sub-optimality was restricted to less than 20% for most cases for all window sizes, note that the "largest" window function ϱ_3 has the fewest cases of highly sub-optimal results.

Table 7.1; informally, the window ϱ_1 results in a multi-resolution cell decomposition with very few significant detail coefficients, (i.e., the environment is represented with high fidelity in a very small neighborhood of the vehicle's location) whereas the windows ϱ_2 and ϱ_3 result in decompositions with progressively larger neighborhoods of high fidelity representation of the environment. We scaled the environment with four different values of D, namely, $D = 6, 7, 8, 9$, and we chose $m_0 = -D$. We performed 30 simulations for each value of D with the initial and goal cells chosen randomly for each simulation, and we executed the multi-resolution path planning algorithm proposed in Section 6.2 with each window function for each simulation (i.e., a total of 120 simulations for each window function).

Figure 7.10 shows the distribution according to percentage sub-optimality of the simulated cases. As intuitively expected, the window ϱ_3 results in the most cases of sub-optimality under 20%. Overall, Fig. 7.10 shows that very few cases of extremely

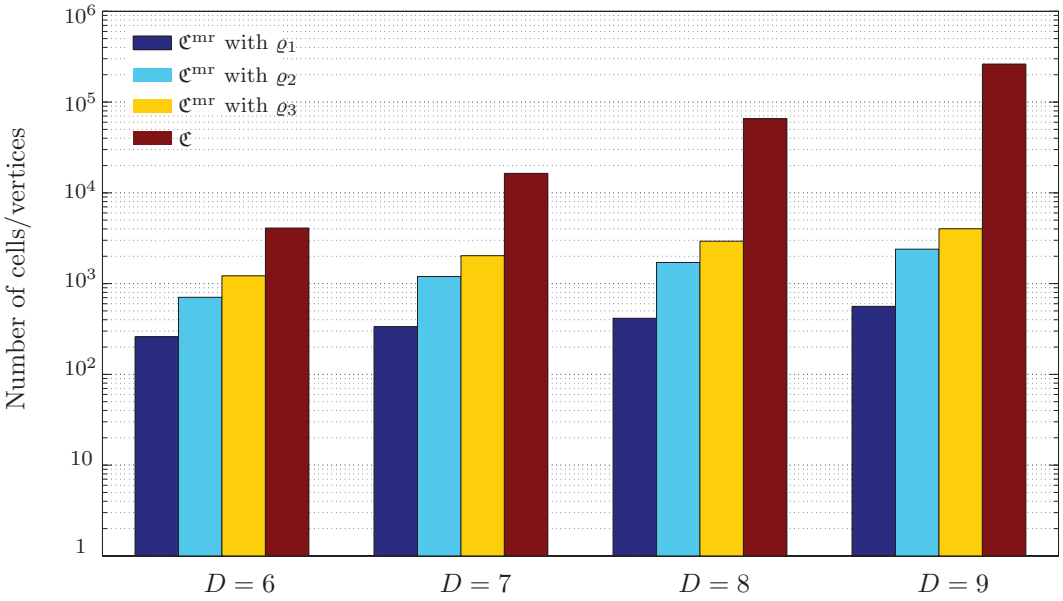


Figure 7.11: Comparison of the number of vertices in the topological graphs associated with multi-resolution cell decompositions with different window functions and with the finest-level cell decomposition $\mathfrak{C}$.

high sub-optimality occurred: these cases typically occurred when the algorithm encountered cul-de-sacs.

Figure 7.11 shows the comparison of the (average) number of vertices in the graphs associated with the multi-resolution cell decompositions corresponding to different window functions and with the finest-level cell decomposition $\mathfrak{C}$. As expected, the window function ϱ_3, which has the largest neighborhood of high fidelity approximation of the environment (i.e., a large number of significant detail coefficients), results in cell decompositions with the largest number of cells among the three multi-resolution decompositions. Note, however, that the numbers of vertices in each of the three multi-resolution decompositions are of the same order of magnitude; on the other hand, the numbers of vertices in the graph $\bar{\mathcal{G}}$ corresponding to the finest-level cell decomposition $\mathfrak{C}$ are one to three orders of magnitude greater than those in the multi-resolution cell decomposition graphs. For instance, with $D = 9$, the number of vertices in $\bar{\mathcal{G}}$ was $262,144$, whereas the average number of vertices was 561 for the multi-resolution cell decomposition with window ϱ_1, $2,395$ with window ϱ_2, and

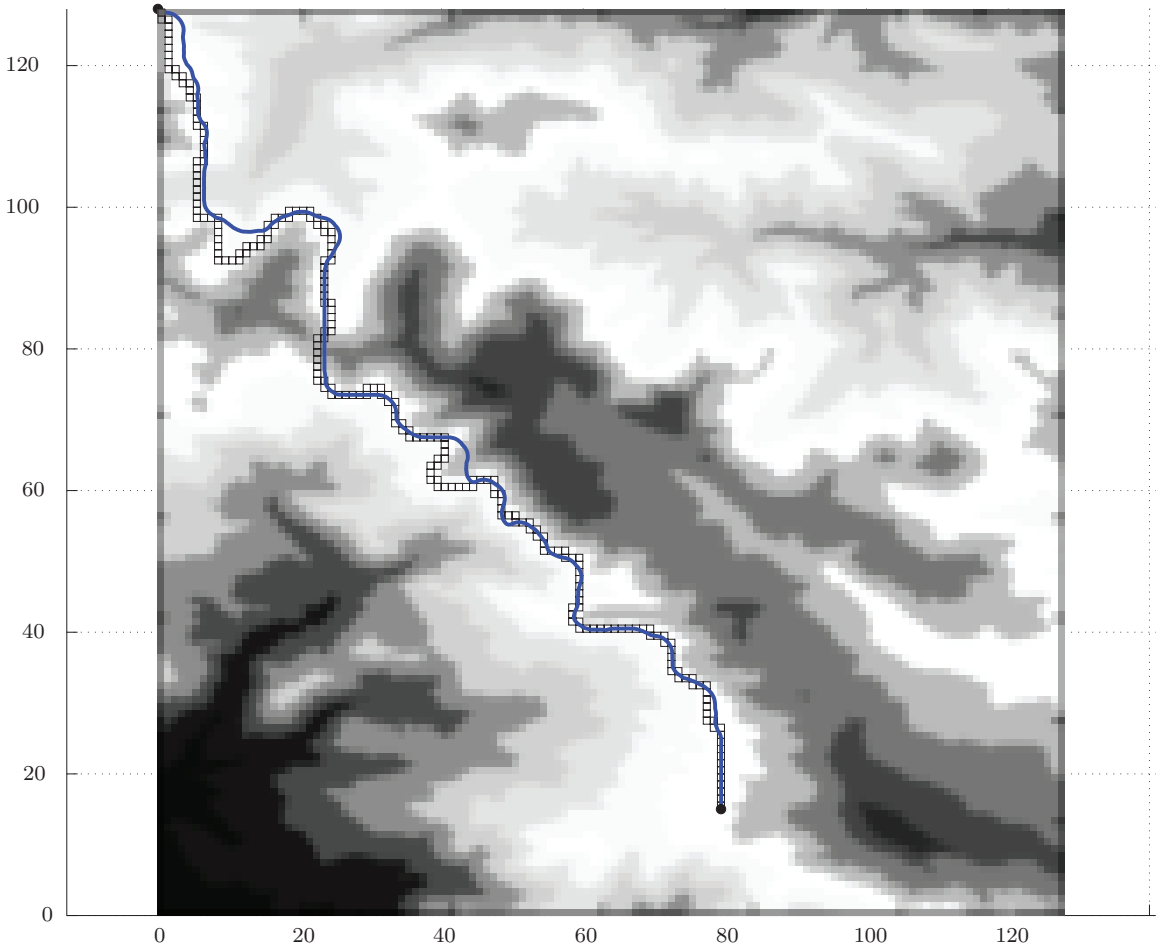
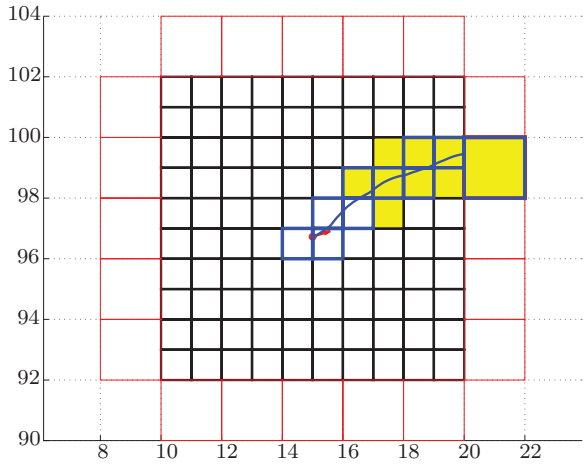


Figure 7.12: Result of motion planning simulation using the aircraft navigational model. The blue curve is the geometric path corresponding to the resultant state trajectory, while the channel of cells in black is the result of executing A* algorithm (without incorporating vehicle dynamical constraints). The initial position is at the top left corner of the map. The units for the x and y axes are kilometers.

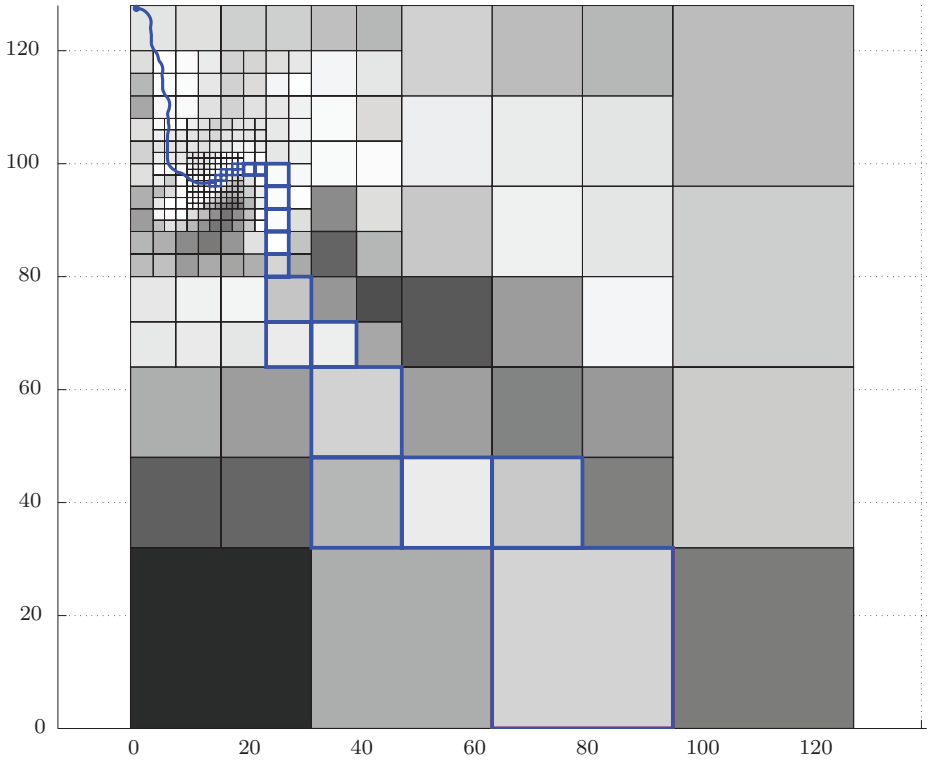
$4,016$ with window ϱ_3. In this context, one may recall that the time complexity of the execution on a sparse graph $\mathcal{G} = (V, E)$ of Dijkstra's algorithm is $O(|V| \ln |V|)$, whereas the memory complexity is $O(|V|)$.

7.3.2 Simulation Results of the Multi-resolution Motion Planning Scheme

Figure 7.12 shows the result of simulating the proposed motion planner for the aircraft navigational model with the following parameters: $C_{\mathrm{D},0} = 0.02$, $K = 0.04$, $S = 30\ \mathrm{m}^2$, $mg = 50$ kN, and $v_{\mathrm{cr}} = 85$ m/s. The aircraft speed was assumed to be constant, and the (asymmetric) bounds on the bank angle control input were $\phi_{\min} = -45°$ and



(a) Intermediate iteration: local perspective. The vehicle's configuration at this iteration is indicated by the red-colored marker and arrow. The x and y axes indicate inertial position coordinates, in kilometers.



(b) Intermediate iteration: global perspective. The multi-resolution cell decomposition represents the environment with high accuracy (small cell sizes) in the immediate vicinity of the vehicle's position. The x and y axes indicate inertial position coordinates, in kilometers.

Figure 7.13: Illustration of an intermediate iteration of the overall motion planning framework.

$\phi_{\max} = 20°$. The objective is to minimize a cost defined on the environment (indicated by regions of different intensities in Fig. 7.12; the darker regions correspond to higher costs).

Figure 7.13 illustrates an intermediate iteration in this simulation example. Figure 7.13(a) shows the cells of size at most $\bar{d}$, with the boundary cells indicated in red. The sequence of cells outlined in blue and the blue-colored curve within this sequence are the results of the H-cost motion planner (the yellow-colored cells indicate the vertices explored during the H-cost search). Figure 7.13(b) shows the overall multi-resolution cell decomposition at the same iteration; the blue-colored cells indicate the optimal path to the goal from the boundary cell chosen by the H-cost motion planner. The blue-colored curve in Fig. 7.13(b) indicates the geometric path traversed by the vehicle in previous iterations. In this simulation example, we chose $\bar{d} = 2$ km.

7.4 Curvature-bounded Traversal of Rectangular Channels

As mentioned in Chapter 5, the constructions of the effective target configuration sets is closely related to the problem of determining the existence of curvature-bounded paths traversing a rectangular channel. In this context, we present a simulation result illustrating the effectiveness of the geometric analysis presented in Chapter 5 and in Appendix C.

Figure 7.15 shows a rectangular channel of non-uniform width. The size of the entire environment is 200 units. First consider the problem of planning a path with a (constant) minimum radius of curvature $r_{\min} = 13$ units. Bereg and Kirkpatrick [14] guarantee the existence of a path of minimum radius of curvature $r_{\min}$ if the (uniform) width of the channel is at least $\tau r_{\min}$, where τ is a constant which satisfies a certain polynomial equation, and $\tau \approx 1.50$. The channel shown in Fig. 7.15 is of non-uniform width, but suppose we consider a uniform width channel lying completely within the given channel. The width d of such a channel is less than or equal to the width of

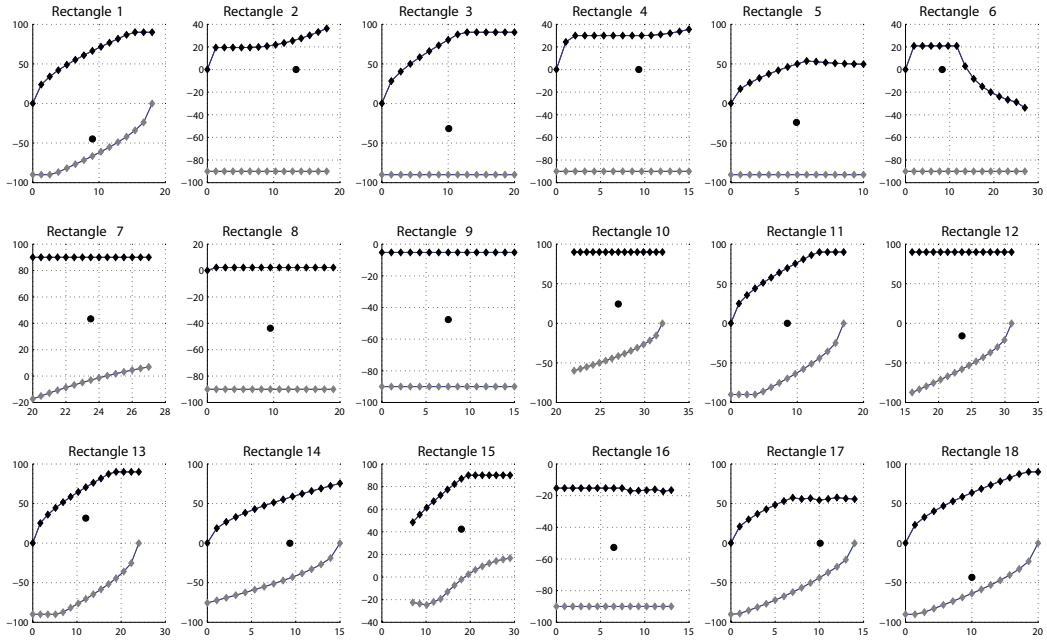


Figure 7.14: Allowable orienations at the entry segments of each rectangle.

the first rectangle, which is chosen as 18 units. Then, the specification of $r_{\min} = 13$ units violates the condition $d \geqslant \tau r_{\min}$. Thus, there is no guarantee that a path with a minimum radius of curvature of 13 units exists within the given channel.

It may be argued that the given rectangular channel may be treated as a special case of a polygonal channel, and the geometric techniques presented in [18] or [2] could be applied. This is true if the specified lower bound on the radius of curvature is *constant*. However, if the specified bounds are different inside different cells, then these methods cannot be applied directly because there is no method to select the terminal conditions within each rectangle.

The proposed approach addresses both these issues. The gray path shown in Fig. 7.15 has a minimum radius of curvature of 13 units, while the black path satisfies the local curvature conditions. Figure 7.14 shows the result of executing the cone analysis algorithm of Section 5.1 for the channel shown in Fig. 7.15. Each of the gray curves indicates the functions $\underline{\alpha}_n$ in degrees, and each of the black curves indicates the functions $\overline{\alpha}_n$ in degrees, measured in the local coordinate system attached to

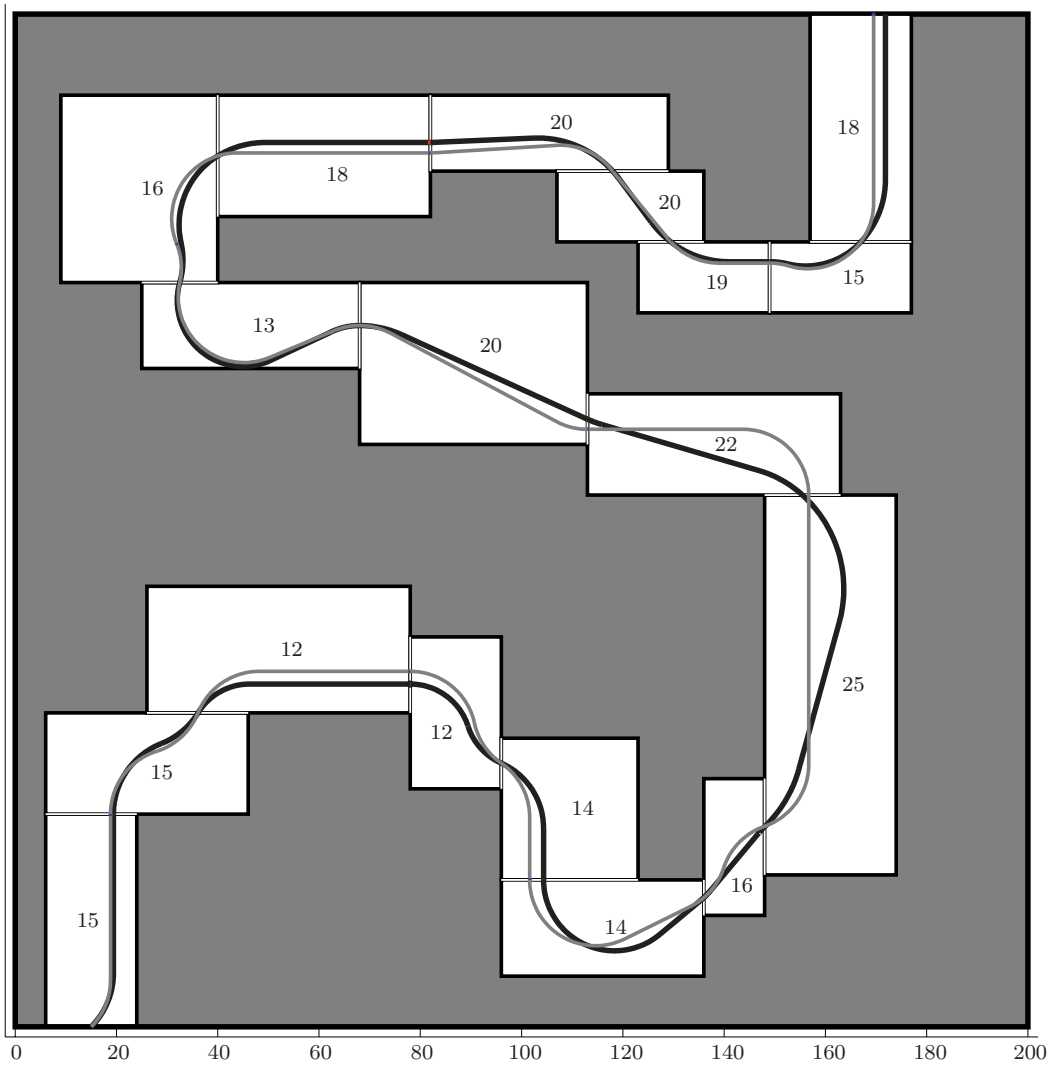


Figure 7.15: Path planning through a channel: The gray path has a constant curvature bound of 13 units, while the black path has variable curvature bound. The numbers within each rectangle indicate local curvature bounds for the latter.

each rectangle. The dots on each plot indicate the initial condition (position and orientation) for each rectangle, measure from the local coordinate system attached to each rectangle. The cone analysis shows that, for any initial condition (position and orientation) at the entry segment of *any* of the intermediate rectangles, there exists a path which satisfies the specified curvature conditions and traverses through the remainder of the channel if and only if the initial condition lies in region enclosed by the upper and lower curves of Fig. 7.14 for that rectangle.

Chapter 8

Conclusions and Directions of Future Work

In this thesis, we discussed a hierarchical motion planning framework that considers vehicle dynamical constraints. The proposed motion planner attempts to find optimal trajectories without sacrificing heavily on its computational efficiency.

To this end, we proposed a novel discretization of the workspace: namely, the lifted graph, which is based on multiple successive edge transitions in the topological graph associated with a workspace cell decomposition. We developed a motion planning framework based on finding optimal paths in the lifted graph, enabled by an efficient and flexible algorithm that we proposed for finding optimal paths in the lifted graph. The edge transition costs in the lifted graphs, i.e., the H-costs, are provided by a local trajectory generation algorithm called TilePlan.

We provided precise and general specifications for the development of TilePlan. In contrast with local trajectory generation problems that arise frequently in other hierarchical approaches to motion planning, the specifications on TilePlan give rise to a *highly structured* problem. We demonstrated the advantages of dealing with a highly structured problem by simplifying the trajectory generation problem using the concept of effective target sets. In particular, the structure of the tile motion planning problem allowed the transformation of non-convex state constraints to convex constraints, via effective target sets.

We addressed the computation of effective target sets based on the observation that vehicle dynamical constraints impose upper bounds on the local curvature of the geometric paths corresponding to feasible state trajectories. We illustrated the precise calculations of these curvature constraints, and we developed geometric analyses

to enable the computation of the effective target sets based on these curvature constraints. The said geometric analyses were noted to enable the solution of a problem of independent interest, namely, the existence of curvature-bounded paths in polygonal environments with a special structure. Using the computation of effective target sets, we provided an implementation of TilePlan for general vehicle models based on model predictive control.

As an example of an efficient discrete path planner, we extended a multi-resolution path planning algorithm previously introduced in the literature. In particular, we modified the existing algorithm to guarantee its completeness. Furthermore, we developed a multi-resolution motion planning scheme that considers vehicle dynamical constraints via H-cost approach in the aforesaid multi-resolution path planner.

The existing literature related to path- and motion planning displays the following traits: computationally efficient techniques that address path optimality (geometric path planners) do not typically consider kinematic/dynamic constraints; computationally efficient techniques that address kinematic/dynamic constraints (randomized sampling-based algorithms) do not typically address optimality; and finally, techniques that address both optimality and kinematic/dynamic constraints are not computationally efficient enough for real-time, online implementations. In this thesis, we addressed the problem of developing the ideal, computationally efficient point-to-point motion planning technique that addresses both optimality and kinematic/dynamic constraints, with the aim of extending easily the proposed technique to the general motion planning problem. Numerical simulation data corroborated our claim that the proposed motion planning framework is a significant, novel contribution towards the development of such an motion planning technique.

In what follows, we outline the possible future extensions of the proposed motion planning framework.

8.1 Motion Planning with Complex Task Specifications

The point-to-point motion planning problem is a special case of the general motion planning problem, in that the search for a feasible path in the cell decomposition graph may be considered as an example of a planning task. As mentioned in Section 1.1, the (discrete) task specification is typically associated with a state transition system, and the satisfaction of the task specification is equivalent to finding a feasible path in this transition system. In the case of the point-to-point problem, this state transition system corresponds directly with the cell decomposition graph.

For the sake of a concrete example, consider a surveillance task where the autonomous vehicle is required to visit a finite number of locations in the environment in a temporally constrained order. For another example, consider the problem of transporting multiple types of cargo between multiple warehouse locations, akin to the classic "man-goat-wolf" problem. Another example, which encompasses the additional aspect of optimality in the general motion planning problem, is the well-known traveling salesperson problem. Depending on the relative locations of the points of interest (i.e., the regions to be surveyed, or the warehouses, in the former two examples) in context with the curvature constraints imposed by the vehicle dynamics, the natural approach of separating the solutions of the task satisfaction problem and the motion control problem may lead to incompatibility issues similar to that illustrated in the motivating example of Chapter 2.

The proposed motion planning framework is currently applicable only to the point-to-point problem due to two restrictions: firstly, we constructed the lifted graph based on multiple edge transitions in the cell decomposition graph (instead of those in a general state transition system), and secondly, we focused on a discrete planner that attempts to find shortest paths in the lifted graph (instead of finding via other means paths satisfying constraints).

It is easy to envision the development of a motion planner that removes these two

restrictions. In particular, one may construct a "lifted transition system" associated with multiple transitions in the transition system associated with the task specification. Note, crucially, that these transitions need not correspond to *cell transitions* because multiple task states may be associated with the same cell. However, the interaction between the two planning levels in the proposed framework occurs due to the consideration of multiple *cell* transitions. The "lifted transition system" will hence need to be designed to capture multiple cell transitions instead of multiple state transitions alone.

Also, finding feasible and/or optimal paths in the natural state transition system associated with a task specification need not be the easiest, or even a practical approach to satisfying the task specification. For example, Kloetzer and Belta [82] discuss the satisfaction of temporal logic specifications by executing a model checking algorithm on the so-called Büchi automaton associated with the cell decomposition graph. The proposed motion planning framework may be extended by developing efficient implementations of such task satisfaction methods in the context of the "lifted transition system".

8.2 Incremental Planning on the Lifted Graph

The search for feasible and/or shortest paths on the lifted graph $\mathcal{G}_H$ (or the "lifted transition system" previously described) is computationally intensive because the number of vertices in the lifted graph grows exponentially with H. For autonomous vehicles with limited on-board computational resources, such as small UAVs, the search for H-cost optimal paths may be infeasible for all but the smallest values of H. Whereas online path *finding* on $\mathcal{G}_H$ may be slow, online path *repair* may be a computationally efficient alternative. Incremental algorithms such as D* and LPA*, discussed in Section A.2 are based on path repair, which refers to the appropriate modification of a previously known path.

Figure 7.12 illustrates that a sequence of cells containing a feasible trajectory may differ only slightly from the sequence of cells found without considering dynamics. This observation can form the basis of an *incremental H-cost search*, described informally as follows: The higher level geometric planner first searches for a path π in the cell decomposition graph without considering the vehicle dynamical constraints. The path π corresponds to a path Π in the lifted graph $\mathcal{G}_H$. It is possible, using TilePlan, to verify if the edges of $\mathcal{G}_H$ contained in the path Π are in fact traversable, considering the vehicle dynamical constraints. In this context, one may envision the implementation of an incremental search algorithm such as LPA* *on the lifted graph* to repair the path Π if some of its edges are found by TilePlan to be infeasible for traversal.

Indeed, this approach of path repair may be used for solving the general motion planning problem. As previously mentioned, it may be computationally intensive/infeasible to search for a path "lifted transition system". However, an incremental algorithm that repairs a path found by the natural hierarchical decomposition of the motion planning problem (e.g., by the geometric planner for the point-to-point problem) may be computationally efficient.

8.3 Generalizations and Extensions of the Proposed Work

8.3.1 Path Planning with Uncertainties

In this thesis, we assumed perfect knowledge of the vehicle's environment map. In practice, the environment map is typically uncertain, and transitions through cells is modeled by a Markov decision process (MDP), and the objective of the path planner is to find a path of minimum *expected* cost. The environment may be modeled by probabilistic occupancy grids, and additionally, the environment model may have a multi-resolution characteristic, similar to the multi-resolution cell decompositions discussed in Chapter 6.

To incorporate vehicle dynamical constraints, the motion planning problem may be modeled by a "lifted MDP," where the states of this "lifted MDP" are sequences of states of the original MDP. As before, the challenge is to extend the methods of solving MDPs to *efficiently* solve the "lifted MDP," as the number of states in the "lifted MDP" will be extremely large. Note however, that in the context of modeling an uncertain environment with probabilistic occupancy grids and the path planning problem by a MDP, only the discrete planner in the proposed motion planner need be changed: the tile motion planning problem remains the same.

The tile motion planning problem *does* change in the presence of uncertainties in the vehicle dynamical model. In this case, the tile planning problem may be posed in terms of *robustness*, i.e., the problem of guaranteeing that the vehicle remains within the specified tile during the tile traversal in the presence of bounded uncertainties in the vehicle dynamical model.

8.3.2 Variable Lengths of Histories

In this thesis, we mainly considered implementations of the proposed H-cost motion planner for fixed values of H. However, it is easy to envision implementations of the proposed motion planner with dynamically changing values of H. Different lengths of histories may be required, for instance, to accommodate vehicle models that exhibit large variations in speed and, consequently, in the minimum radii of turn: agile rotorcraft, for example. In such cases, variable lengths of histories may allow for an effective trade-off between the computational efficiency of the planner and the optimality of the resultant path: whereas longer histories better represent the vehicle dynamical constraints, H-cost searches with (uniformly) large values of H are slow.

The implementation of an H-cost motion planner with variable H is not difficult. In fact, we have already illustrated one such implementation, namely, the multi-resolution motion planner based on the partially lifted graph in Chapter 6. There,

the lengths of histories were varied according to the sizes of the cells; appropriate definitions of history lengths varying according to the vehicle speed/minimum radius of turn may be accommodated similarly in an H-cost search algorithm with variable lengths of histories.

The more important and challenging problem is of choosing appropriately the lengths of histories corresponding to the vehicle's speed and/or the (local) minimum radius of turn. In this thesis, we resorted to trial-and-error for selecting H; automatic selection of H will be an important generalization of the proposed work.

8.3.3 Planar Motion with Larger Configuration Spaces

The idea of attaching history-dependent costs in workspace cell decomposition graphs is closely related to *differential flatness*. The states and inputs of a differentially flat system can be expressed in terms of the so-called *flat outputs* and their derivatives. In this context, the idea of attaching history-dependent costs may be thought of as a method of recovering derivative information about the vehicle's position. Due to flatness, this derivative information allows the planner to calculate the orientation by the relation $\theta = \tan^{-1}(\dot{y}/\dot{x})$.

In light of the preceding observation, it is conceivable to extend the proposed framework for motion planning of all differentially flat vehicle models with the position coordinates (of some point on the vehicle, not necessarily the center of mass) as the flat outputs. For example, the kinematical model of a car with n trailers is differentially flat [129] with the position coordinates of the last trailer as the flat outputs.

The extension of the proposed motion planner to such vehicle models may be achieved by solving appropriately the tile motion planning problem. The flatness property suggests that the tile motion planning problem may at least *in principle* be solvable, as all the other configuration variables of the vehicle model may be extracted from the history of the position coordinates (flat outputs).

Appendix A

Theoretical Background

A.1 Optimal Control

Optimal control theory, which can be traced historically as far back as the 17^{th} century to the brachistochrone problem, addresses the following problem:

Find an admissible control input u^* and an admissible state trajectory ξ^* that minimize the performance measure $\int_0^{t_\text{f}} \ell(\xi(t;\xi_0,u),u,t)\ \mathrm{d}t$, subject to the differential constraint $\dot{\xi} = f(\xi,u)$.

The works of Kirk [80], Athans and Falb [6], and Bertsekas [15] provide comprehensive introductions to optimal control theory. Here we review briefly and qualitatively the major ideas involved in the solution of the problem stated above.

The central result of the *variational theory* of optimal control is the famous *Pontyagin Minimum Principle* (PMP). Informally, the PMP states that the optimal control input u^* and the corresponding state trajectory x^* minimize pointwise in time a scalar function of the state and the input, called the *Hamiltonian*, defined by

$$\mathcal{H}(\xi,u,p,t) := \ell(\xi,u,t) + p^{\mathrm{T}} f(\xi,u),$$

where p, called the *co-state*, is a variable of the same dimension as the state. The PMP is a necessary, but not sufficient, condition of optimality. The PMP identifies a family of control inputs to which belongs an optimal control input; to find specifically an optimal control input – i.e., to solve the so-called *synthesis problem* – it is usually required to solve the following two-point boundary value problem associated with the minimization of the Hamiltonian:

$$\dot{\xi^*}(t) = \frac{\partial \mathcal{H}}{\partial p}(t), \qquad\qquad \dot{p^*}(t) = -\frac{\partial \mathcal{H}}{\partial \xi}(t),$$

with $\xi(0) = \xi_0$. The terminal boundary values of the co-state are given by the so-called *transversality conditions*. Whereas the synthesis problem can be solved analytically in special cases (cf. [144]), for the large majority of optimal control problems, the preceding boundary value problem is solved numerically.

The preceding two-point boundary value problem is related to the synthesis of an *open-loop* optimal control input for a specific initial condition. The design of an optimal *feedback policy* is based on the *Bellman Principle of Optimality*, stated informally as follows:

A trajectory $t \mapsto \xi^*(t)$, $t \in [0, t_\mathrm{f}]$, *with* $\xi^*(0) = \xi_0$ *to* $\xi^*(t_\mathrm{f}) = \xi_\mathrm{f}$ *is an optimal trajectory from* ξ_0 *to* ξ_f *if and only if the trajectory* $s \mapsto \xi^*(s)$, $s \in [t, t_\mathrm{f}]$ *is an optimal trajectory from* $\xi^*(t)$ *to* ξ_f, *for all* $t \in [0, t_\mathrm{f}]$.

The Bellman Principle of Optimality leads to the following partial differential equation, called the *Hamilton-Jacobi-Bellman equation* (HJB), for obtaining the so-called *value function* J^*:

$$-\frac{\partial J^*}{\partial t}(\xi, t) \;=\; \min_u \{\mathcal{H}(\xi, u, \frac{\partial J^*}{\partial \xi}(\xi, t), t)\},$$

with the boundary condition $J^*(\xi_\mathrm{f}, t_\mathrm{f}) = 0$. An optimal control u^* may then be synthesized, *for any initial state* ξ, by computing a minimizer of $\mathcal{H}(\xi, u, \frac{\partial J^*}{\partial \xi}(\xi, t), t)$.

A.2 Classical Planning

Classical planning refers to the search for a sequence of actions leading to a pre-specified goal. The problem is assumed to be fully observable (the current state is known precisely), time invariant (the effects of actions do not change over time), and deterministic. Classical planning problems are specified using a so-called *domain description language*, the most widely known example of which is the STRIPS language. These languages typically specify task states as a conjunction of atomic logical propositions (called *literals*). An *action* is specified by a *pre-condition* that determines the

set of task states to which this action may be applied, and an *effect* that specifies the successor state when this action is applied to a particular task state.

Classical planning problems are associated naturally with a directed graph in which task states are associated with the vertices and actions are associated with the edges of the graph. Planning approaches may be broadly classified as either a *state space search* or a *plan space search*. The former class of approaches refers to searching for a feasible and/or optimal (in terms of the number of actions) path in the aforesaid directed graph. The reader interested is referred to the works of Nilsson [112] and Russell and Norvig [130] for extensive surveys of solutions to the classical planning problem.

A.2.1 Search Algorithms

A *search algorithm* involves exploring new states (i.e., exploring new vertices of the aforesaid graph by traversing its edges) until the goal state is explored. different search algorithms are characterized according to the strategy used for exploring new states. *Uninformed search* refers to a search strategy that assumes no knowledge of the problem other than the problem definition itself. Well-known uninformed search algorithms include the *breadth-first search* (vertices explored the earliest are chosen for further exploration), the *depth-first search* (the latest vertices explored are chosen for further exploration), and *uniform cost search* (the vertices with least known cost are chosen for further exploration). The well-known Dijsktra search algorithm is a uniform cost, uninformed search.

Informed search refers to a search strategy that has *a priori* knowledge about the problem that enables a ranking of newly explored vertices in terms of their "closeness" to the goal. Informed searches are associated with a heuristic function that guides the search along the "most promising" vertices. The A* algorithm is a well-known example of an informed search.

Search algorithms are characterized by four fundamental properties: soundness, completeness, efficiency, and optimality. The *soundness* of an algorithm refers to the validity of its result, i.e., whether every possible result of the algorithm is a feasible sequence of actions leading to the goal. The *completeness* property refers to an algorithm's ability to find a solution to the planning problem if there exists one, and to terminate in a finite number of iterations otherwise. The *efficiency* of an algorithm is further characterized by its *time complexity* and its *space complexity*, which refer, respectively, to the computational time and memory requirements of the algorithm in relation to the number of states and the number of actions. Finally, the *optimality* of an algorithm refers to its capability of finding sequences of actions consisting of the minimum number of actions.

A.2.2 Incremental Search

The classical planning problem assumes full observability and time-invariance. However, some states and/or actions may not be known *a priori* and/or may change during the course of planning. In the context of geometric path planning for example, the actual environment map may be different from the map known *a priori*. In this case, all changes in the environment are captured by changes in the vertex set and/or the edge set and/or the edge transition costs in the cell decomposition graph that represents the environment. Consequently, a change in the environment necessitates a new graph search to find a path commensurate with that change. However, this process may be computationally expensive and wasteful, especially when the change in the environment is local, i.e., restricted to small region(s) of the environment. It is desirable to re-use information about the previous optimal path for computing the new optimal (considering the change in the environment), i.e., avoid recomputing the optimal path "from scratch."

To this end, the so-called *incremental* graph search algorithms find the new optimal path without discarding information about the previously known optimal path. The research in incremental graph search algorithms was pioneered by Stentz [142,143] with the invention of the D* algorithm; later references discussing incremental algorithms include Refs. [70, 83–85, 98, 120].

A.3 Hierarchical and Hybrid Systems

Autonomous vehicles in general, and motion planning and control systems in particular, are examples of systems involving both discrete and continuous subsystems, typically in a hierarchical control structure. The behavior of purely discrete systems has been addressed in the fields of discrete event systems (cf. [25]) and artificial intelligence (cf. [130]; also, the behavior of purely continuous systems has been thoroughly addressed in the field of dynamical systems and control theory (cf. [5, 75]). However, the study of *hybrid* systems is a recent, emerging discipline.

Hybrid control systems are typically designed in a hierarchical structure, and it is important to characterize the behavior of hierarchical systems with respect to the behaviors of the individual subsystems involved.

A.3.1 Theory of Hierarchical Systems

We discuss the concepts of *coordinability* and *consistency* for hierarchical systems, introduced in the work of Mesarović *et al* [107].

A *system* is defined as a function $S : U \to Y$ between a set of inputs U and a set of outputs Y. In particular, a *decision-making system* chooses a *decision* x from a decision set X such that x is the result of a *decision problem*. The output of a decision-making system is a function $\phi : X \to Y$ of the decision. Two general decision problems may be identified:

1. The system S is said to solve an *optimization problem* if the decision x corresponding to an input $u \in U$ minimizes a given objective function that depends,

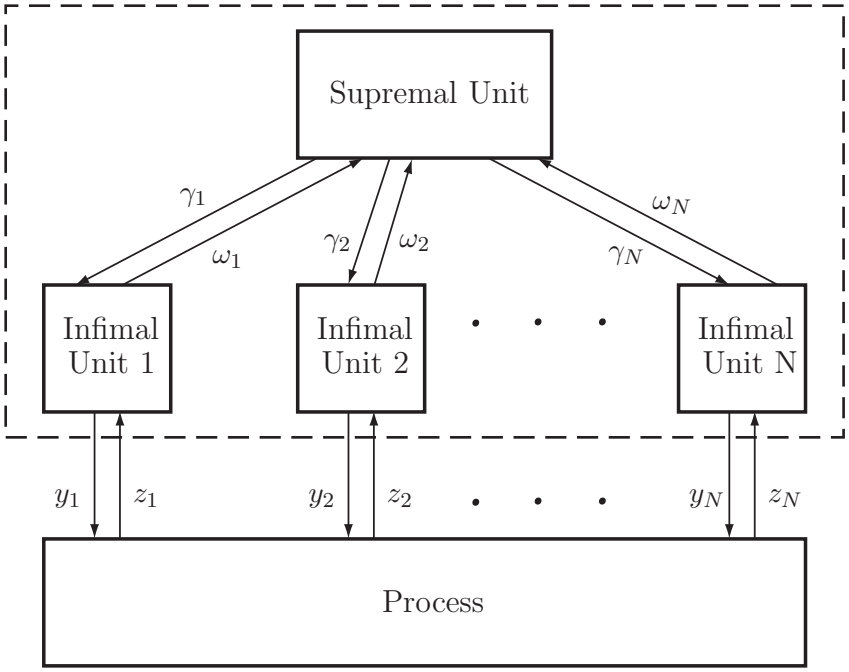


Figure A.1: Two-level hierarchical control of a process.

in general, on both x and u.

2. The system S is said to solve a *satisfaction problem* if the decision x corresponding to an input $u \in U$ satisfies a given constraint function that depends, in general, both on x and u.

A two-level hierarchy of decision-making systems, consisting of a single higher-level unit and N lower-level units and controlling a certain process, is shown in Fig. A.1. In accordance with the terminology used in [107], we refer to the higher level decision-making unit as the *supremal unit* and to the lower level decision-making units as the *infimal units.*

We denote by $\mathfrak{D}_{\mathrm{S}}$ the supremal decision problem, and by $\mathfrak{D}_{\mathrm{I},i}$ the decision problem of the i^{th} infimal unit. Correspondingly, we denote by $\gamma = (\gamma_1, \ldots, \gamma_N)$ the supremal decision, and by x_i the i^{th} infimal decision. The output of the i^{th} infimal unit is $y_i = \phi_i(x_i)$.

We acknowledge explicitly the dependency of the infimal decision problem on the

supremal decision by denoting by $\mathfrak{D}_{\mathrm{I},i}(\gamma_i)$ the i^{th} infimal decision problem. For the sake of brevity, we denote the set of infimal decision problems by $\overline{\mathfrak{D}}_{\mathrm{I}}(\gamma)$ and the set of infimal decisions by x, i.e., $\overline{\mathfrak{D}}_{\mathrm{I}}(\gamma) = \{\mathfrak{D}_{\mathrm{I},1}(\gamma_1), \ldots, \mathfrak{D}_{\mathrm{I},N}(\gamma_N)\}$, and $x = \{x_1, \ldots, x_N\}$. Similarly, we denote the set of outputs of the infimal units by $y = \phi(x)$. Finally, we denote by $\overline{\mathfrak{D}}$ the overall decision problem of the two-level system. We are now ready to define the notions of coordinability and consistency.

The infimal decision problems $\mathfrak{D}_{\mathrm{I},i}$ are said to be *coordinable relative to the supremal decision problem* [107, pp. 94] if the following proposition is true:

$$(\exists\gamma)(\exists x)\left[\mathbf{P}(x, \overline{\mathfrak{D}}_{\mathrm{I}}(\gamma)) \text{ and } \mathbf{P}(\gamma, \mathfrak{D}_{\mathrm{S}})\right].$$

Informally, coordinability relative to the supremal decision problem means that it is possible for the supremal unit to solve its decision problem in a manner such that it is in turn possible for the infimal units to solve *their* decision problems (which, in general, depend on the solution of the supremal decision problem).

Next, we introduce a complementary notion of coordinability. The infimal decision problems are said to be *coordinable relative to the overall decision problem* [107, pp. 95] if the following proposition is true:

$$(\exists\gamma)(\exists x)\left[\mathbf{P}(x, \overline{\mathfrak{D}}_{\mathrm{I}}(\gamma)) \text{ and } \mathbf{P}(\phi(x), \overline{\mathfrak{D}})\right].$$

Informally, coordinability relative to the overall decision problem means that if each of the infimal units solves its decision problem (which depends on the supremal decision), then the overall decision problem is solved.

Finally, the infimal and supremal decision problems are said to be *consistent* if the following proposition is true:

$$(\forall\gamma)\,(\forall x)\left[\mathbf{P}(x, \overline{\mathfrak{D}}_{\mathrm{I}}(\gamma)) \Rightarrow \mathbf{P}(\phi(x), \overline{\mathfrak{D}})\right] \tag{A.1}$$

The proposition (A.1) is called the *consistency postulate* [107, pp. 97]. Informally, consistency means that the infimal decision units are coordinated relative to

the overall decision problem whenever they are are coordinated relative to the supremal decision problem. In other words, the two-level system depicted in Fig. A.1 is consistent if the overall system objective is achieved when the supremal unit and each of the infimal units achieve their own objectives.

The preceding definitions characterize precisely the *interactions* between the different subsystems in a hierarchical control structure. In particular, the consistency postulate provides a precise mathematical definition of the notion of "harmonious" operation of a hierarchical system.

A.3.2 Hybrid Systems Analysis and Control

Hybrid systems are systems involving both discrete and continuous state variables. Owing to their inherently multi-disciplinary nature, hybrid systems have been studied by different academic communities for different applications; consequently, there is no single, universal model of hybrid systems. For example, Alur *et al* [3] introduce the *hybrid automaton*, where the transitions of a finite state automaton are governed by the evolution of continuous variables; Branicky *et al* [21] introduce a model consisting of dynamical system with controlled and uncontrolled jumps in the state, such that the behavior of the system is governed by different dynamical equations in different regions of the state space; and Goebel et al [49] discuss a differential inclusion model of hybrid systems. Introductory texts and surveys in hybrid systems theory include Refs. [35, 49, 50, 97, 146, 150].

From a control theoretic perspective, the problems of *stability analysis* (cf. [20, 49, 97, 161]) and *robust and optimal control* (cf. [21, 49]) are of interest in the context of hybrid systems. The results on stability typically focus on constructing Lyapunov-like functions for hybrid systems, and on the extension of known results for (continuous) nonlinear systems to hybrid systems. In the context of hybrid optimal control, Sussmann [145] and Piccoli [117] provide necessary conditions of optimality, similar to

the Pontryagin Minimum Principle; whereas Hedlund and Rantzer [52, 53] consider the solutions of Bellman recurrence-type equations for hybrid systems via dynamic programming.

From a computer scientific perspective, the problem of *verification* of hybrid systems, which refers to the problem of deciding whether the hybrid system can reach a pre-specified set of discrete states from a pre-specified initial state, is of interest. In purely discrete systems, verification is a matter of efficient implementation alone, because in principle, the set of reachable states from a given initial state can be determined by enumeration (the number of state trajectories is finite). Verification in hybrid systems, however, poses the additional problem of *decidability*, i.e., the question of whether verification can be achieved at all. The main idea involved in the verification of hybrid systems is of *discrete abstraction* [4, 146], which refers to the approximation by a discrete state model of the continuous dynamics. The "closeness" to the continuous system of this discrete approximation is characterized by *equivalence properties* such as simulation, bisimulation, language inclusion, and language equivalence. The construction of such "equivalent" models of continuous systems has been addressed, for instance, in [4, 51, 115, 146, 149].

A.4 The Discrete Wavelet Transform

Multi-resolution analysis of a scalar function of one variable is the construction of a hierarchy of functional approximations by projecting the function onto a sequence of nested linear spaces. The discrete wavelet transform provides a framework for such multi-resolution analysis (MRA) of a function. In this framework, the sequence of nested linear spaces is generated by translated and scaled versions of two functions $\phi : \mathbb{R} \to \mathbb{R}$ and $\psi : \mathbb{R} \to \mathbb{R}$ of unit energy, called the *scaling function* and *wavelet* respectively. The scaling function and the wavelet must be chosen such that they

satisfy the *orthogonality equations* for each $n \in \mathbb{Z}$:

$$\langle \phi(t), \phi(t-n) \rangle = \delta(n), \qquad \langle \psi(t), \psi(t-n) \rangle = \delta(n), \qquad \langle \psi(t), \phi(t-n) \rangle = 0, \quad \text{(A.2)}$$

and such that there exist sequences $h(n)$ and $g(n)$ of scalars satisfying the *dilation equations*

$$\phi(t) = \sum_{n=-\infty}^{\infty} h(n)\phi(2t-n), \qquad \psi(t) = \sum_{n=-\infty}^{\infty} g(n)\phi(2t-n). \qquad \text{(A.3)}$$

For each $m, k \in \mathbb{Z}$, we define the functions $\phi_{m,k} : \mathbb{R} \to \mathbb{R}$ and $\psi_{m,k} : \mathbb{R} \to \mathbb{R}$ by

$$\phi_{m,k} := \sqrt{2^m}\phi(2^m t - k), \qquad \psi_{m,k} := \sqrt{2^m}\psi(2^m t - k).$$

For each $m \in \mathbb{Z}$, let $\mathcal{V}_m$ be a linear subspace of $\mathbb{L}^2(\mathbb{R})$ defined as $\mathcal{V}_m := \mathsf{span}\left\{\{\phi_{m,k} : k \in \mathbb{Z}\}\right\}$. It can be shown [121, Ch. 3] that $\mathcal{V}_{m-1} \subset \mathcal{V}_m$, for each $m \in \mathbb{Z}$; that $\bigcup_m \mathcal{V}_m$ is dense in $\mathbb{L}^2(\mathbb{R})$; and that $\{\psi_{m,k}\}$ is a basis set for $\mathcal{W}_m := \mathcal{V}_m \backslash \mathcal{V}_{m-1}$.

The *discrete wavelet transform* of a function $\mathbb{L}^2(\mathbb{R}) \ni f : \mathbb{R} \to \mathbb{R}$ is defined by

$$a_{m_0,k} := \langle \phi_{m_0,k}(t), f(t) \rangle, \qquad d_{m,k} := \langle \psi_{m,k}(t), f(t) \rangle, \qquad \text{(A.4)}$$

where $m_0 \in \mathbb{Z}$ is pre-specified. The corresponding *reconstruction equation* is

$$f(t) \;=\; \sum_{k=-\infty}^{\infty} a_{m_0,k}\phi_{m_0,k}(t) + \sum_{m=m_0}^{\infty}\sum_{k=-\infty}^{\infty} d_{m,k}\psi_{m,k}(t). \qquad \text{(A.5)}$$

The scalars $a_{m_0,k}$ and $d_{m,k}$ are known as *approximation* and *detail* coefficients respectively. The first term in the R.H.S. of (A.5) is the approximation of f *at resolution* m_0, while the inner summation of the second term is the difference between approximations at two successive levels of resolution.

The 2-D wavelet transform is an extension of the 1-D wavelet transform, where a scaling function and three wavelets are defined by

$$\Phi_{m,k,\ell}(x,y) := \phi_{m,k}(x)\phi_{m,\ell}(y), \quad \Psi^1_{m,k,\ell}(x,y) := \phi_{m,k}(x)\psi_{m,\ell}(y),$$
$$\Psi^2_{m,k,\ell}(x,y) := \psi_{m,k}(x)\phi_{m,\ell}(y), \quad \Psi^3_{m,k,\ell}(x,y) := \psi_{m,k}(x)\psi_{m,\ell}(y).$$

For each $m \in \mathbb{Z}$, let $\mathcal{V}_m, \mathcal{W}_m^1, \mathcal{W}_m^2$, and $\mathcal{W}_m^3$ be subspaces defined as follows:

$$\mathcal{V}_m := \mathsf{span}\left\{\{\Phi_{m,k,\ell}(x,y) : k,\ell \in \mathbb{Z}\}\right\}, \qquad \mathcal{W}_m^i := \mathsf{span}\left\{\left\{\Psi_{m,k,\ell}^i(x,y) : k,\ell \in \mathbb{Z}\right\}\right\}.$$

Note that the difference $\mathcal{V}_m \backslash \mathcal{V}_{m-1}$ is equal to the union of the mutually orthogonal spaces $\mathcal{W}_m^1, \mathcal{W}_m^2, \mathcal{W}_m^3$. The *2-D discrete wavelet transform* of a function $\mathbb{L}^2(\mathbb{R}^2) \ni F : \mathbb{R}^2 \to \mathbb{R}$ is

$$\begin{aligned} a_{m_0,k,\ell} &:= \langle \Phi_{m_0,k,\ell}(x,y), F(x,y)\rangle, \\ d_{m,k,\ell}^i &:= \langle \Psi_{m,k,\ell}^i(x,y), F(x,y)\rangle, \qquad \text{for } i = 1,2,3, \quad m \geqslant m_0, \quad k,\ell \in \mathbb{Z}, \end{aligned}$$

where $m_0 \in \mathbb{N}$ is pre-specified. The corresponding *2-D reconstruction equation* is

$$F(x,y) = \sum_{k,\ell=-\infty}^{\infty} a_{m_0,k,\ell}\Phi_{m,k,\ell}(x,y) + \sum_{i=1}^{3}\sum_{m=m_0}^{\infty}\sum_{k,\ell=-\infty}^{\infty} d_{m,k,\ell}^i \Psi_{m,k,\ell}^i(x,y). \quad \text{(A.6)}$$

An example of a pair of scaling function and wavelet is the Haar family, defined by

$$\phi(t) := \begin{cases} 1 & 0 \leqslant t < 1 \\ 0 & \text{otherwise} \end{cases}, \qquad \psi(t) := \begin{cases} 1 & 0 \leqslant t < 1/2 \\ -1 & 1/2 \leqslant t < 1 \\ 0 & \text{otherwise.} \end{cases} \quad \text{(A.7)}$$

For the 1-D Haar family, the subspace $\mathcal{V}_m$, $m \in \mathbb{Z}$, corresponds to the set of piecewise-constant functions over the regularly spaced, disjoint intervals $I_{m,k} := [2^{-m}k, 2^{-m}(k+1)]$ of length 2^{-m}, for $k \in \mathbb{Z}$. The approximation coefficient $a_{m,k}$ is equal to the average value of the function over the interval $I_{m,k}$. Consequently, for the 2-D Haar family, the subspace $\mathcal{V}_m$, $m \in \mathbb{Z}$ corresponds to the set of piecewise-constant functions over the square regions

$$S_{m,k,\ell} := I_{m,k} \times I_{m,\ell} = \left[2^{-m}k, 2^{-m}(k+1)\right] \times \left[2^{-m}\ell, 2^{-m}(\ell+1)\right] \quad \text{(A.8)}$$

of size 2^{-m}, for $k,\ell \in \mathbb{Z}$. Accordingly, the approximation coefficient $a_{m,k,\ell}$ is equal to the average value of the function over the square region $S_{m,k,\ell}$.

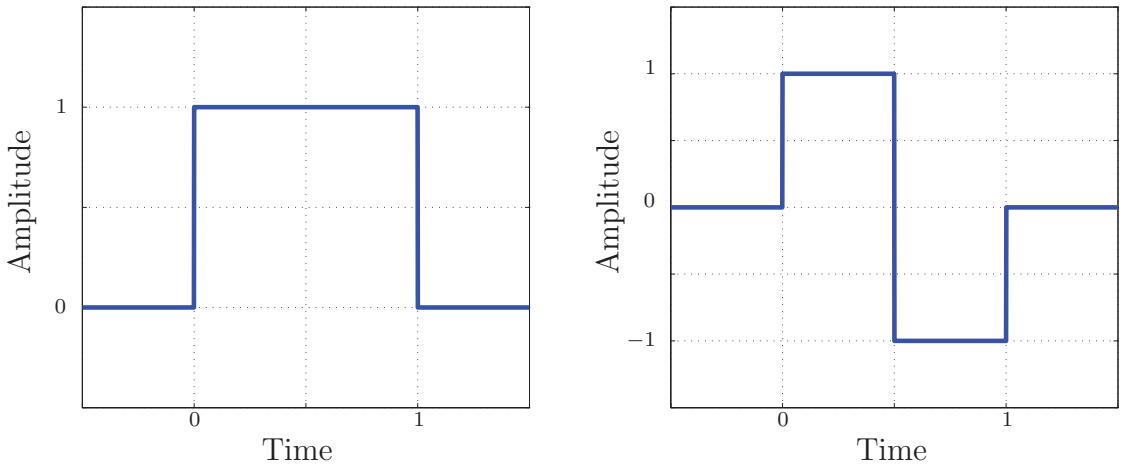


Figure A.2: The 1-D Haar scaling function and wavelet functions.

The discrete wavelet transform is a powerful and computationally efficient tool widely used in multi-resolution signal processing [37, 106, 121]. Several recent works have investigated the applications of wavelet transforms to vision-based navigation and vision-based SLAM: see, for instance, Ref. [104] (appearance-based vision-only SLAM); Refs. [62] and [28] (wavelet analysis for local feature extraction); and Ref. [138] (stereo image processing). With the plethora of available sensors, and in light of the fact that multiple sensors are typically used for autonomous navigation [148], the wavelet transform may soon become the common standard of the representation and analysis of signals [31]. In this context, several recent works address wavelet-based data representation: see, for instance, Ref. [163] (occupancy grids); Ref. [155] (standardized representation of road roughness characteristics); Ref. [156] (terrain depiction for pilot situational awareness); and Ref. [116] (image registration using wavelets for vision-based navigation).

Appendix B

Technical Proofs

B.1 Technical Results for Chapters 2 and 3

Proof of Proposition 2.4. We first show that the algorithm terminates after a finite number of iterations. Whenever a pair (j, m) is added to $\mathcal{P}$, the value of $d(j, m)$ is reduced. Since there is a finite number of nodes and T_H is finite, there is also a finite number of paths containing no cycles from i_{S} to i. Since the number of reductions of $d(j, m)$ is at most equal to the number of possible paths from i_{S} to i, it follows that the number of possible reductions of $d(j, m)$ is also finite. Hence, each pair (j, m) is added to $\mathcal{P}$ a finite number of times. Since a (j, m) pair is always deleted from $\mathcal{P}$ at each iteration, it follows that $\mathcal{P} = \varnothing$ after a finite number of iterations and the algorithm terminates.

Next, suppose there exists at least one admissible path in $\mathcal{G}$ from i_{S} to i containing a particular history $I :=\textsc{History}(i, m_P)$. Since there are finitely many paths from i_{S} to i with no cycles, and since cycles have non-negative cost, there exists an optimal path. Suppose $\pi^* := (j_0, j_1, \ldots, j_P)$ is an optimal path in $\mathcal{G}$ with $j_0 = i_{\mathrm{S}}$, $j_P = i$, and $(j_{P-H}, \ldots, j_P) = \textsc{History}(i, m_P)$. Also, let m_k be such that $(j_{k-H}, \ldots, j_k) = \textsc{History}(j_k, m_k)$, for $k \in \{H, \ldots, P\}$. Since π^* is optimal, the cost from $j_0 = i_{\mathrm{G}}$ to node $j_k \in \pi^*$ must be $\mathcal{J}^*_{j_k, m_k}$, for every $k \in \{1, \ldots, P\}$.

Next, for sake of contradiction, suppose $d(i, m_P) > \mathcal{J}^*_{i, m_P}$ after the algorithm terminates. This implies that $d(j_{P-1}, m_{P-1}) > \mathcal{J}^*_{j_{P-1}, m_{P-1}}$, for otherwise, when (j_{P-1}, m_{P-1}) were removed from $\mathcal{P}$, the condition in Line 8 of procedure Main would have been satisfied due to the optimality of π^*. Furthermore, Line 9 of procedure

MAIN would have resulted in

$$\begin{aligned} d(i, m_P) &= d(j_{P-1}, m_{P-1}) \\ &\quad + \tilde{g}_{H+1}(([h(j_{P-1}, m_{P-1})]_2, I)) \\ &= \mathcal{J}^*_{j_{P-1}, m_{P-1}} + \tilde{g}_{H+1}(([h(j_{P-1}, m_{P-1})]_2, I)) \\ &= \mathcal{J}^*_{j_P, m_P}, \end{aligned}$$

since $[h(j_{P-1}, m_{P-1})]_2 = j_{P-H-1}$ by Line 10. By similar arguments, $d(j_{P-1}, m_{P-1}) > \mathcal{J}^*_{j_{P-1}, m_{P-1}}$ implies that $d(j_{P-2}, m_{P-2}) > \mathcal{J}^*_{j_{P-2}, m_{P-2}}$ and so on, leading to the conclusion that $d(j_{H+1}, m_{H+1}) > \mathcal{J}^*_{j_{H+1}, m_{H+1}}$. However, Line 6 of procedure INITIALIZE precludes this situation, thus leading to a contradiction. Hence, we must have $d(i, m_P) = \mathcal{J}^*_{i, m_P}$ after the termination of the algorithm. □

Proof of Proposition 2.5. Let $\mathcal{P}_k$ denote the fringe after k iterations of the algorithm. Let also $d_k^{j,m}$ denote the value of $d(j, m)$, $(j, m) \in \mathcal{P}_k$, at this instant. Let

$$\begin{aligned} b_k &:= \min\{d_k^{j,m} : (j, m) \in \mathcal{P}_k\}, \\ (j_k, m_k) &:= \arg\min\{d_k^{j,m} : (j, m) \in \mathcal{P}_k\}. \end{aligned}$$

We claim that $b_0 \leqslant b_1 \ldots \leqslant b_k$, for each $k \geqslant 1$. To this end, note that $b_0 = \min\{d_0^{j,m} : (j, m) \in \mathcal{P}_0\} \geqslant 0$ by Line 6 of procedure INITIALIZE. At initialization, either $|\mathcal{P}_0| = 1$ or $|\mathcal{P}_0| > 1$. First, suppose that $|\mathcal{P}_0| = 1$. Then, after the first iteration is performed for all neighbors of j_0, (j_1, m_1) must be such that $(j_0, j_1) \in E$, and hence

$$b_1 = d_1^{j_1, m_1} = d_0^{j_0, m_0} + \tilde{g}_{H+1}(([h(j_0, m_0)]_2, I)) \geqslant b_0,$$

where $I = \text{HISTORY}(j_1, m_1)$. If $|\mathcal{P}_0| > 1$, then after the first iteration is performed for all neighbors of j_0, $(j_1, m_1) \in \mathcal{P}_1 \backslash \mathcal{P}_0$ or $(j_1, m_1) \in \mathcal{P}_0 \backslash \{(j_0, m_0)\}$. In either case, $b_1 \geqslant b_0$.

Next, assume that $b_0 \leqslant b_1 \leqslant \ldots \leqslant b_k$ for some $k \geqslant 1$. Let $\hat{\mathcal{P}}_{k+1}$ be the set of elements added to $\mathcal{P}$ during the k^{th} iteration, i.e., $\mathcal{P}_{k+1} = (\mathcal{P}_k \backslash \{(j_k, m_k)\}) \cup \hat{\mathcal{P}}_{k+1}$. Note that $d_{k+1}^{j,m} \geqslant d_k^{j_k,m_k} = b_k$ for every $(j,m) \in \hat{\mathcal{P}}_{k+1}$. It follows that

$$\begin{aligned} b_{k+1} &= \min\left\{d_{k+1}^{j,m} : (j,m) \in \mathcal{P}_{k+1}\right\} \\ &= \min\{d_{k+1}^{j,m} : (j,m) \in (\mathcal{P}_k \backslash \{(j_k,m_k)\}) \cup \hat{\mathcal{P}}_{k+1}\} \\ &= \min\{\min\left\{d_{k+1}^{j,m} : (j,m) \in \mathcal{P}_k \backslash \{(j_k,m_k)\}\right\}, \\ &\qquad \min\{d_{k+1}^{j,m} : (j,m) \in \hat{\mathcal{P}}_{k+1}\}\}. \end{aligned} \tag{B.1}$$

Now, for every $(j,m) \in \mathcal{P}_k \backslash \{(j_k,m_k)\}$, we have

$$d_{k+1}^{j,m} = \min\{d_k^{j,m}, d_k^{j_k,m_k} + \tilde{g}_{H+1}(([h(j_k,m_k)]_2^{\cdot} I))\},$$

where $I = \textsc{History}(j,m)$. Since $b_k = d_k^{j_k,m_k} = \min\left\{d_k^{j,m} : (j,m) \in \mathcal{P}_k\right\}$ and since $\tilde{g}_{H+1}$ is non-negative, we have $d_{k+1}^{j,m} \geqslant d_k^{j_k,m_k}$ for every $(j,m) \in \mathcal{P}_k \backslash \{(j_k,m_k)\}$. It follows from (B.1) that $b_{k+1} \geqslant b_k$, and thus by induction we have that $b_0 \leqslant b_1 \leqslant \ldots \leqslant b_k$ for $k \geqslant 1$.

Finally, we show that once a pair (i,m) is removed from $\mathcal{P}$, it is never added back again. To see this, suppose (i,m) is removed after the κ^{th} iteration of the algorithm, i.e., $d_{i,m}^{\kappa-1} = b_{\kappa-1}$. For sake of contradiction, suppose (i,m) re-enters $\mathcal{P}$ after the λ^{th} iteration, with $\lambda > \kappa$. It follows from Line 9 of procedure `Main` that

$$d_\lambda^{i,m} = d_{\lambda-1}^{j_\lambda,m_\lambda} + \tilde{g}_{H+1}(([h(j_\lambda,m_\lambda)]_2, I_{i,m})) \geqslant d_{\lambda-1}^{j_\lambda,m_\lambda} = b_{\lambda-1}.$$

By (8), if (i,m) enters $\mathcal{P}$ then $d_m(i)$ is strictly reduced. It follows that $d_{\lambda-1}^{i,m} > d_\lambda^{i,m} \geqslant b_{\lambda-1}$. However, since $d_{i,m}^k$ is non-increasing, we have $b_{\lambda-1} < d_{\lambda-1}^{i,m} \leqslant d_{\kappa-1}^{i,m} = b_{\kappa-1}$. If $\lambda > \kappa$, then $b_{\lambda-1} \geqslant b_{\kappa-1}$, thus leading to a contradiction. □

Proof of Lemma 3.1. For a given $H \in \mathbb{N}$, the cost $\mathcal{J}_H(\Pi^H)$ of the path Π^H is computed by executing TilePlan for each tile in Π^H, i.e., for each pair $(J_m^{\Pi^H}, J_{m+1}^{\Pi^H})$, $m = 0, 1, \ldots, P-H-1$. By definition of the lifted graph $\mathcal{G}_H$, the edge $(J_m^{\Pi^H}, J_{m+1}^{\Pi^H})$ in $\mathcal{G}_H$ is

a node in V_{H+1}, and the tile $(J_m^{\Pi^{H+1}}, J_{m+1}^{\Pi^{H+1}})$ in $\mathcal{G}_{H+1}$ is the triplet $(J_m^{\Pi^H}, J_{m+1}^{\Pi^H}, J_{m+2}^{\Pi^H})$. Thus, the first node of the tile $(J_m^{\Pi^{H+1}}, J_{m+1}^{\Pi^{H+1}})$ in $\mathcal{G}_{H+1}$ is the same as the first node of the tile $(J_m^{\Pi^H}, J_{m+1}^{\Pi^H})$ in $\mathcal{G}_H$.

Let TilePlan(H) and TilePlan$(H+1)$ denote, respectively, the execution of TilePlan on tiles in $\mathcal{G}_H$ and on tiles in $\mathcal{G}_{H+1}$. Let ξ be the initial state provided as an input to both TilePlan(H) and TilePlan$(H+1)$. Suppose TilePlan(H) computes $u_{m,H}$ as the control required to traverse the tile $(J_m^{\Pi^H}, J_{m+1,H})$, and TilePlan$(H+1)$ computes $u_{m,H+1}$ as the control required to traverse the tile $(J_m^{\Pi^{H+1}}, J_{m+1}^{\Pi^{H+1}})$. Correspondingly, let $\xi_{m,H}$ and $\xi_{m,H+1}$ be the terminal states, respectively, resulting after the controls $u_{m,H}$ and $u_{m,H+1}$ are applied at ξ.

Because TilePlan computes the controls required for traversal of the first cell, and because the first cell of both the tiles under consideration is the same, $u_{m,H+1}$ will differ from $u_{m,H}$ only if there exists no control from the state $\xi_{m,H}$ that would enable traversal through the remainder of the tile $(J_{m,H+1}, J_{m+1,H+1})$, i.e., through the sequence of cells $J_m^{\Pi^{H+1},(3)}, \ldots, J_m^{\Pi^{H+1},(H+3)}$. As a consequence, TilePlan(H) would return failure when it processes the tile $(J_{m+1}^{\Pi^H}, J_{m+2}^{\Pi^H})$, since the first $H+1$ cells of this tile are precisely $J_m^{\Pi^{H+1},(3)}, \ldots, J_m^{\Pi^{H+1},(H+3)}$. In other words, for every tile $(J_m^{\Pi^H}, J_{m+1}^{\Pi^H})$, $m = 0, 1, \ldots, P-H$, if the cost returned by TilePlan(H) is different from the cost returned by TilePlan$(H+1)$ for the tile $(J_m^{\Pi^{H+1}}, J_{m+1}^{\Pi^{H+1}})$, then TilePlan$(H)$ must return an infinite cost (failure) for the tile $(J_{m+1}^{\Pi^H}, J_{m+2}^{\Pi^H})$. The required result (3.7) then follows. □

B.2 Technical Results for Chapter 6

Proposition B.1. *Let $\bar{j} \in \bar{V}$, and $\mathcal{A} =$ MR-Approx$(\bar{j})$. Let $\mathfrak{C}^{\mathrm{mr}}$ and $\mathcal{G} = (V, E)$ be, respectively, the multi-resolution cell decomposition and the topological graph associated with $\mathcal{A}$. If there exists an obstacle-free path in $\bar{\mathcal{G}}$ from $\bar{j}$ to $\bar{i}_{\mathrm{G}}$, then there exists an obstacle-free path in $\mathcal{G}$ from $j := \mathsf{vert}(\mathsf{cell}(\bar{j}; \mathfrak{C}^{\mathrm{mr}}); \mathcal{G})$ to i_{G}, where $i_{\mathrm{G}} \in V$ is*

the unique node that satisfies $\bar{i}_{\mathrm{G}} \in W(i_{\mathrm{G}}, V)$.

Proof. Let $\bar{\pi}(\bar{j}, \bar{i}_{\mathrm{G}}) = (\bar{j}_0, \ldots, \bar{j}_{\bar{P}})$ be an obstacle-free path in $\bar{\mathcal{G}}$ from $\bar{j}_0 = \bar{j}$ to $\bar{j}_{\bar{P}} = \bar{i}_{\mathrm{G}}$. For each $m = 0, 1, \ldots, \bar{P}$, there exists a unique set $W_m \in \{W(j, V)\}_{j \in V}$ such that $\bar{j}_m \in W_m$. Let $i_m \in V$ be such that $W_m = W(i_m, V)$. Because $\bar{\pi}$ is a path in $\bar{\mathcal{G}}$, $(\bar{j}_{m-1}, \bar{j}_m) \in \bar{E}$ for each $m = 1, 2, \ldots, \bar{P}$, and it follows that either $W_{m-1} = W_m$, or $(i_{m-1}, i_m) \in E$. Thus, the path $\pi(j, i_{\mathrm{G}}) := \{j_0, \ldots, j_P\}$, where $P \leqslant \bar{P}$, is a path in $\mathcal{G}$.

To show that the path π is also obstacle-free in $\mathcal{G}$, we note that since $\bar{\pi}$ is obstacle-free in $\bar{\mathcal{G}}$, $F(\bar{j}_m) \leqslant 1 - \varepsilon$, for each $m = 0, 1, \ldots, \bar{P}$. It follows by (6.8) that $\hat{F}(\mathsf{cell}(j_m; \mathfrak{C}^{\mathrm{mr}})) < (1 - \varepsilon)$ for each $m = 0, 1, \ldots, P$, and by (6.10) that $\mathcal{J}(\pi) < M$, i.e., π is an obstacle-free path. □

Corollary B.2. *If there exists an obstacle-free path in* $\bar{\mathcal{G}}$ *from the initial node* $\bar{i}_{\mathrm{S}}$ *to the goal node* $\bar{i}_{\mathrm{G}}$*, then the cost of the initial path* π_0^* *computed by the algorithm is finite.*

Proof. By Proposition B.1, if there exists an obstacle-free path in $\bar{\mathcal{G}}$ from $\bar{j}$ to $\bar{i}_{\mathrm{G}}$, then there exists an obstacle-free path $\pi_0^*(i_{\mathrm{S}}, i_{\mathrm{G},0})$ in $\mathcal{G}(0)$ from the node $i_{\mathrm{S}} := \mathsf{vert}(\mathsf{cell}(\bar{i}_{\mathrm{S}}; \mathfrak{C}); \mathcal{G}(0))$ to the node $i_{\mathrm{G},0}$, where $i_{\mathrm{G},0} \in V(0)$ is the unique node that satisfies $\bar{i}_{\mathrm{G}} \in W(i_{\mathrm{G},0}, V(0))$. Because π_0^* is obstacle-free, $\mathcal{J}(\pi_0^*) < M$, i.e., $\mathcal{J}(\pi_0^*)$ is finite. □

Proposition B.3. *Suppose that the algorithm does not meet a setback at iteration* $n \in \mathbb{N}$ *of its execution, and also suppose that* $\textsc{Visited}(\bar{j}_n) = 0$*. If there exists a path in the graph* $\mathcal{G}(n)$ *from the node* $j_n = \mathsf{vert}(\mathsf{cell}(\bar{j}_n; \mathfrak{C}^{\mathrm{mr}}(n)); \mathcal{G}(n))$ *to the node* $i_{\mathrm{G},n}$*, then* $\mathcal{K}_{\mathrm{G}}(\bar{j}_{n-1}) - \mathcal{K}_{\mathrm{G}}(\bar{j}_n) \geqslant \lambda_2$*, where* $i_{\mathrm{G},n} \in V(n)$ *is the unique node that satisfies* $\bar{i}_{\mathrm{G}} \in W(i_{\mathrm{G},n}, V(n))$.

Proof. Let $\pi_n^*(j_n, i_{\mathrm{G},n}) = (j_0, \ldots, j_{P(n)})$ denote the optimal path in the graph $\mathcal{G}(n)$ computed by the algorithm at Line 11. First, suppose that the cell decomposition $\mathfrak{C}^{\mathrm{mr}}(n)$ is identical to the cell decomposition $\mathfrak{C}^{\mathrm{mr}}(n-1)$ (in particular, $i_{\mathrm{G},n-1} = i_{\mathrm{G},n}$). If

there exists a path in $\mathcal{G}(n)$ from j_n to $i_{\mathrm{G},n}$, then there exists an optimal path in $\mathcal{G}(n)$ from j_n to $i_{\mathrm{G},n}$ because $\mathcal{G}(n)$ is finite. Then, by Bellman's principle of optimality, the path $\pi^*_{n-1}(j_{n-1}, i_{\mathrm{G},n-1}) = (i_0, \ldots, i_{P(n-1)})$, computed at iteration $n-1$ of the algorithm, contains the path π^*_n, with $P(n) = P(n-1) - 1$, and $j_{m-1} = i_m$ for each $m = 1, 2, \ldots, P(n)$, and hence $\mathcal{J}(\pi^*_n) \leqslant \mathcal{J}(\pi^*_{n-1})$.

Next, suppose that the cell decomposition $\mathfrak{C}^{\mathrm{mr}}(n)$ is not identical to the cell decomposition $\mathfrak{C}^{\mathrm{mr}}(n-1)$. Let $\pi_n(j_n, i_{\mathrm{G},n})$ and $\pi_{n-1}(j_{n-1}, i_{\mathrm{G},n-1})$ be paths in the graphs $\mathcal{G}(n)$ and $\mathcal{G}(n-1)$ respectively. If $\mathcal{W}(\pi_n) \subseteq \mathcal{W}(\pi_{n-1})$, then due to the second and third terms in the right hand side of (6.10), $\mathcal{J}(\pi_n) \leqslant \mathcal{J}(\pi_{n-1})$. In particular, if $\mathcal{W}(\pi^*_n) \subseteq \mathcal{W}(\pi^*_{n-1})$, then $\mathcal{J}(\pi^*_n) \leqslant \mathcal{J}(\pi^*_{n-1})$.

Now suppose $\mathcal{W}(\pi^*_n) \nsubseteq \mathcal{W}(\pi^*_{n-1})$. Let $\pi_n(j_n, i_{\mathrm{G},n})$ be any path in $\mathcal{G}(n)$ from u_n to $i_{\mathrm{G},n}$ satisfying $\mathcal{W}(\pi_n) \subseteq \mathcal{W}(\pi^*_{n-1})$. There exists at least one such path π_n in $\mathcal{G}(n)$ because the algorithm does not meet a setback at iteration n. By the arguments in the preceding paragraph, $\mathcal{J}(\pi_n) \leqslant \mathcal{J}(\pi^*_{n-1})$. Furthermore, because π^*_n is an optimal path in $\mathcal{G}(n)$ from j_n to $i_{\mathrm{G},n}$, $\mathcal{J}(\pi^*_n) \leqslant \mathcal{J}(\pi_n)$, and it follows that $\mathcal{J}(\pi^*_n) \leqslant \mathcal{J}(\pi^*_{n-1})$.

Finally, note that the cell corresponding to the first node $j_0 \in V(n)$ in the path π^*_n is the same as the cell corresponding to the second node $i_1 \in V(n-1)$ in π^*_{n-1}, and furthermore, this cell corresponds to the node $\bar{j}_n \in \bar{V}$. Then $\mathcal{K}_{\mathrm{G}}(\bar{j}_{n-1}) - \mathcal{K}_{\mathrm{G}}(\bar{j}_n) = \mathcal{J}(\pi^*_{n-1}) - \mathcal{J}(\pi^*_n) \geqslant \bar{g}(\bar{j}_{n-1}, \bar{j}_n) \geqslant \lambda_2$, by (6.9). □

Proposition B.4. *Let $\bar{j}$ be an arbitrary node in $\bar{V}$. Then either the algorithm never visits $\bar{j}$ or the algorithm visits $\bar{j}$ finitely many times.*

Proof. Suppose, for the sake of contradiction, that the algorithm visits the node $\bar{j} \in \bar{V}$ infinitely many times at iterations $n_1, n_2, \ldots, n_k \ldots$, i.e., $\bar{j}_{n_1} = \bar{j}_{n_2} = \ldots = \bar{j}$. By Line 8, $\mathcal{K}_{\mathrm{G}}(\bar{j}_{n_k}) - \mathcal{K}_{\mathrm{G}}(\bar{j}_{n_{k-1}}) > 0$, and hence there exists $N \in \mathbb{N}$, such that $\mathcal{K}_{\mathrm{G}}(\bar{j}_{n_N}) \geqslant M$. It follows by Line 3 that the algorithm terminates in at most n_N iterations, leading to a contradiction. □

Proposition B.5. *Let $\pi_n^*(j_n, i_{\mathrm{G},n}) = (j_0, \ldots, j_{P(n)})$ be the path found by the algorithm either at Line 8 or Line 11 at iteration $n \in \mathbb{N}$, and suppose there exists an obstacle-free path in $\bar{\mathcal{G}}$ from $\bar{j}_n$ to $\bar{i}_{\mathrm{G}}$ that is contained within the set $\mathcal{W}(\pi_n^*)$. Then the algorithm does not visit the node $\bar{j}_n$ at any future iteration.*

Proof. We note that the cell corresponding to the second node in the path π_n^* is a cell at the finest resolution, and hence, $W(j_1, V(n)) = \bar{j}_{n+1}$. Then it follows due to (6.11) and due to the hypothesis that there exists an obstacle-free path $\bar{\pi}(\bar{j}_n, \bar{i}_{\mathrm{G}}) = (\bar{i}_0, \ldots, \bar{i}_{\bar{P}})$ in $\bar{\mathcal{G}}$ from $\bar{j}_n$ to $\bar{i}_{\mathrm{G}}$ such that $\bar{i}_1 = \bar{j}_{n+1}$. Thus, there exists an obstacle-free path in $\bar{\mathcal{G}}$ from $\bar{j}_{n+1}$ to $\bar{i}_{\mathrm{G}}$: in particular, $(\bar{i}_1, \ldots, \bar{i}_{\bar{P}})$ is such a path. Then it follows by Proposition B.1 that the algorithm does not execute Line 18 at iteration $n+1$.

By the preceding arguments, the following statement is true: if there exists an obstacle-free path in $\bar{\mathcal{G}}$ from $\bar{j}_{n+k}$ to $\bar{i}_{\mathrm{G}}$ contained within π_{n+k}^*, then the algorithm does not execute Line 18 at iteration $n+k+1$.

Now suppose, for the sake of contradiction, that there exists $\ell > 1$ such that the algorithm visits node $\bar{j}_n$ again at iteration $n+\ell$, i.e., $\bar{j}_n = \bar{j}_{n+\ell}$ and $\bar{j}_{n+1} = \bar{j}_{n+\ell-1}$. Then there exists $m < \ell$ such that for each $k = m, m+1, \ldots, \ell$, the algorithm executes Line 18 at iteration $n+k$, i.e. $\bar{j}_{n+k+1} = b(\bar{j}_{n+k})$. Due to the statement in the preceding paragraph, it follows that either there exists no obstacle-free path in $\bar{\mathcal{G}}$ from $\bar{j}_{n+k}$ to $\bar{i}_{\mathrm{G}}$, or the second node of every obstacle-free path in $\bar{\mathcal{G}}$ from $\bar{j}_{n+k}$ to $\bar{i}_{\mathrm{G}}$ is $b(\bar{j}_{n+k})$. However, neither of these hold true for $k = \ell - 1$, because we showed earlier that $(\bar{i}_1, \ldots, \bar{i}_{\bar{P}})$ is an obstacle-free path in $\bar{\mathcal{G}}$ from $\bar{j}_{n+1} = \bar{j}_{n+\ell-1}$ to $\bar{i}_{\mathrm{G}}$, and this path does not contain $\bar{j}_n$. Thus we arrive at a contradiction, and it follows that there exists no $\ell > 1$ such that $\bar{j}_n = \bar{j}_{n+\ell}$, i.e., the algorithm does not visit $\bar{j}_n$ at any future iteration. □

Appendix C

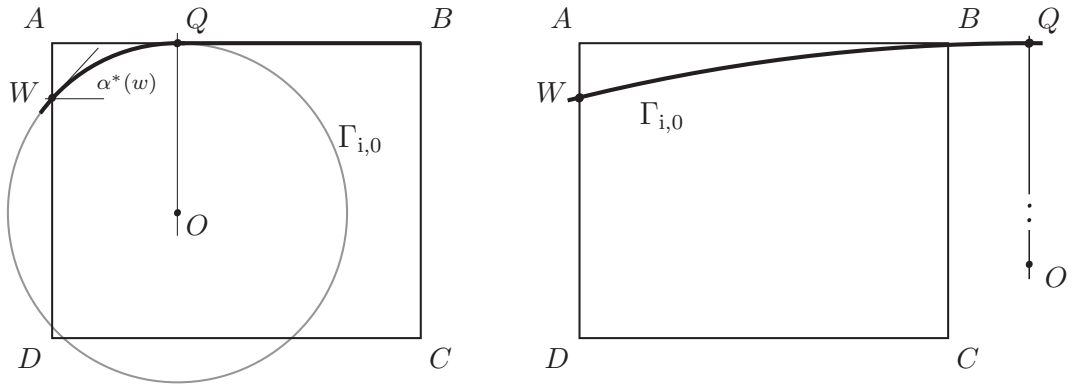
Existence of Curvature-bounded Paths in Rectangular Channels: General Case

As discussed in Chapter 5, our solutions of Problems 4.4 and 4.5 are based on constructions of paths Λ_x and Υ_x satisfying properties (P1)-(P5). We describe these constructions in this chapter. For our constructions, we only consider paths that are concatenations of circular arcs of radius r and straight line segments, which is justified by the following result.

Lemma C.1 (Boissonnat *et al* [18])**.** *If there exists an admissible*[1] *path of curvature at most* r^{-1}*, then there exists an admissible path consisting of a concatenation of straight line segments and arcs of circles of radius* r*.*

In what follows, we will denote by C^+ a clockwise circular arc, by C^- a counter-clockwise circular arc, and by S a straight line segment. When necessary, we will denote by C_u^+, C_u^-, or S_u an arc of length u. Also, we will denote by C^+C^+, C^+C^-, C^-S, or $C_u^+C_v^+$, $C_u^+C_v^-$, $C_u^-S_v$, etc. concatenations of different arcs. It is implicit that at the points of concatenation of successive sections of a path, the tangents of both sections are equal, thus ensuring that the overall path remains continuously differentiable. In particular, when we refer to a line of a specified slope being tangent to a specified clockwise arc or a specified counter-clockwise arc, the point of tangency is uniquely determined.

[1] *In [18], the term "admissible" defines a curvature-bounded continuously differentiable path that satisfies specified initial and terminal conditions and is contained within a specified polygon.*



(a) $r \leqslant \dfrac{d_1^2 + (d_2 - w)^2}{2(d_2 - w)}$ (b) $r > \dfrac{d_1^2 + (d_2 - w)^2}{2(d_2 - w)}$

Figure C.1: Maximum possible tangent angle at W of a Type 1 admissible path.

C.1 Traversal across Parallel Edges

We first establish a series of preliminary results characterizing the geometry of curvature-bounded paths involved in the traversal across parallel edges.

Lemma C.2. *If* $r \leqslant \dfrac{d_1^2 + (d_2 - w)^2}{2(d_2 - w)}$, *then the maximum possible tangent angle at* W *for any Type 1 admissible path is given by the function* $w \mapsto \alpha^*(w)$, *defined by*

$$\alpha^*(w) \;:=\; \begin{cases} \frac{\pi}{2}, & d_2 - w > r, \\ \cos^{-1}\left(1 - \dfrac{d_2 - w}{r}\right), & d_2 - w \leqslant r. \end{cases} \tag{C.1}$$

Proof. Let $\alpha \in [0, \frac{\pi}{2}]$, and let $\Gamma_{i,0}$ be the C^+ arc of radius r, passing through W such that $\Gamma'_{i,0}(W) = \alpha$. Let O be the center of the circular arc $\Gamma_{i,0}$, and let Q be the intersection with $\Gamma_{i,0}$ of the line that passes through O and is parallel to the y axis (see Fig. C.1(a)). It follows from elementary geometry that $\Gamma_{i,0}$ either does not intersect the line AB or it intersects the line AB in at most one point if and only if

$$r(1 - \cos\alpha) \;\leqslant\; d_2 - w. \tag{C.2}$$

First, suppose $d_2 - w/r > 1$. Then (C.2) is satisfied for all $\alpha \in [0, \frac{\pi}{2}]$, and the maximum possible value of $\Gamma'_{i,0}(W)$ is $\frac{\pi}{2}$. Next, suppose $d_2 - w/r \leqslant 1$. Then $\alpha^*(w)$

satisfies (C.2) as an equality, and the point Q lies on the segment AB (see Fig. C.1(a)). The abscissa of $Q = (q, d_2)$ is

$$q := r \sin \alpha^*(w) = \sqrt{2r(d_2 - w) - (d_2 - w)^2} \leqslant d_1,$$

since $r \leqslant \dfrac{d_1^2 + (d_2 - w)^2}{2(d_2 - w)}$. It follows that Q lies between points A and B, and hence $\alpha^*(w)$ equals the maximum possible value of $\Gamma'_{\mathrm{i},0}(W)$, for otherwise if $\Gamma'_{\mathrm{i},0}(W) > \alpha^*(w)$, then $\Gamma_{\mathrm{i},0}$ would intersect the segment AB in two points and $\Gamma_{\mathrm{i},0}$ would not be Type 1 admissible.

It remains to show that no Type 1 admissible path other than a C^+ arc passing through W can have an initial tangent angle greater than $\alpha^*(w)$. If $d_2 - w > r$, then $\alpha^*(w) = \frac{\pi}{2}$, and there is nothing to prove. Suppose $d_2 - w \leqslant r$, in which case the C^+ arc $\Gamma_{\mathrm{i},0}$ passing through W with $\Gamma'_{\mathrm{i},0}(W) = \alpha^*(w)$ intersects the segment AB at exactly one point Q, and $\Gamma'_{\mathrm{i},0}(Q) = 0$. Note that the absolute rate of change of orientation along a continuously differentiable path is by definition the curvature of the path. In the family of paths with curvature at most r^{-1} everywhere, a circle of radius r is the path where the absolute rate of change of orientation is maximum, since the curvature along the circle is equal to r^{-1} everywhere. It follows that any continuously differentiable path Π from W to Q with $\Pi'(W) > \alpha^*(w)$ and $\Pi'(Q) \leqslant 0$ necessarily has curvature greater than r^{-1} at some point on that curve. Now, for the sake of contradiction, suppose there exists a continuously differentiable path Π between W and Q such that $\Pi'(W) > \alpha$ and $\Pi'(Q) > 0$, and such that the curvature at all points on Π is at most r^{-1}. By continuity of the tangent, Π intersects $\Gamma_{\mathrm{i},0}$ at a point P between W and Q. Then $\Pi'(W) > \Gamma'_{\mathrm{i},0}(W)$ (by assumption) and $\Pi'(P) < \Gamma'_{\mathrm{i},0}(P)$, which implies that the curvature of the section of Π between W and P exceeds r^{-1} at some point on that section of Π. □

Corollary C.3. *Let $X = (d_1, x)$ be any point on segment YZ, and suppose $r \leqslant \dfrac{d_1^2 + (d_2 - x)^2}{2(d_2 - x)}$. If $d_2 - x < r$ and $\overline{\beta}(x) < -\cos^{-1}(1 - (d_2 - x)/r)$, then there*

exists no Type 1 admissible path between W and X.

Corollary C.4. *Let $X = (d_1, x)$ be any point on segment YZ, and suppose $r \leqslant \dfrac{d_1^2 + x^2}{2x}$. If $x < r$ and $\underline{\beta}(x) > \cos^{-1}(1 - x/r)$, then there exists no Type 1 admissible path between W and X.*

Corollaries C.3 and C.4 are direct consequences of Lemma C.2 after reflections about the horizontal and vertical axes, respectively. In light of these Corollaries C.3 and C.4, a preliminary step common to all Cases is to determine, for a particular $x \in [y, z]$, whether the hypotheses of these corollaries hold true, i.e., whether $\overline{\beta}(x) < -\cos^{-1}(1 - (d_2 - x)/r)$ whenever $r \leqslant (d_1^2 + (d_2 - x)^2)/2(d_2 - y)$ and $(d_2 - x)/r < 1$ hold; or whether $\underline{\beta}(x) \leqslant \cos^{-1}(1 - x/r)$ whenever $r \leqslant (d_1^2 + x^2)/2x$ and $x < r$ hold. If either hypothesis holds true, then we conclude that Υ_x does not exist *for that particular* $x \in [y, z]$. For the sake of clarity, we will assume in what follows that neither of the above conditions holds true.

Fact C.5. *Let X be a point on the segment YZ, and let $\beta \in \left[\underline{\beta}(x), \overline{\beta}(x)\right]$ be fixed. If there exists a Type 1 admissible C^+SC^- (resp. C^-SC^+) path between (W, α) and (X, β), then there exists no Type 1 admissible C^+SC^+ (resp. C^-SC^-) path between (W, α) and (X, β).*

Proof. Let $\Gamma_{\mathrm{i},0}$ be the C^+ arc passing through W such that $\Gamma'_{\mathrm{i},0}(W) = \alpha$. Let δ_{C^+S} be the angle of slope of the line passing through X that is tangent to $\Gamma_{\mathrm{i},0}$. Suppose $\beta < \delta_{C^+S}$, and let $\Gamma_{\mathrm{f},1}$ be the C^- arc passing through X with $\Gamma'_{\mathrm{f},1}(X) = \beta$. Then the C^+SC^- path formed by concatenating sections of $\Gamma_{\mathrm{i},0}$ and $\Gamma_{\mathrm{f},1}$ (along with a section of the common tangent to $\Gamma_{\mathrm{i},0}$ and $\Gamma_{\mathrm{f},1}$) is not Type 1 admissible because the point of intersection with $\Gamma_{\mathrm{f},1}$ of the common tangent to $\Gamma_{\mathrm{i},0}$ and $\Gamma_{\mathrm{f},1}$ lies outside the rectangle $ABCD$.

On the other hand, if $\beta \geqslant \delta_{C^+S}$, then this C^+SC^- path is Type 1 admissible, while the C^+SC^+ path formed by concatenating $\Gamma_{\mathrm{i},0}$ and $\Gamma_{\mathrm{f},0}$ is not Type 1 admissible due

to the same argument as above. Since α and β are fixed, any C^+SC^- path (resp. C^+SC^+ path) between (W,α) and (X,β) must be consist of sections of $\Gamma_{\mathrm{i},0}$ and $\Gamma_{\mathrm{f},1}$ (resp. $\Gamma_{\mathrm{f},0}$), and the result follows. □

Fact C.6. *Let X be a point on the segment YZ, and let $\beta \in \left[\underline{\beta}(x), \overline{\beta}(x)\right]$ be fixed. If there exists a Type 1 admissible C^+SC^+ (resp. C^-SC^-) path between (W,α) and (X,β), then there exists no Type 1 admissible C^-SC^- (resp. C^+SC^+) path between (W,α) and (X,β).*

Proof. Similar to the proof of Fact C.5, hence omitted. □

Let $\mathfrak{D}$ be the set of all paths consisting of at most three sections excluding the $C^+C^-C^+$ and $C^-C^+C^-$ paths, i.e.,

$$\mathfrak{D} := \left\{ C^+_{\ell_1} S_{\ell_2} C^+_{\ell_3},\ C^+_{\ell_1} S_{\ell_2} C^-_{\ell_3},\ C^-_{\ell_1} S_{\ell_2} C^+_{\ell_3},\ C^-_{\ell_1} S_{\ell_2} C^-_{\ell_3} : \ell_1, \ell_2, \ell_3 \in \mathbb{R}_{\geqslant 0} \right\}.$$

Lemma C.7. *Let X be a point on the segment YZ, and let $\beta \in \left[\underline{\beta}(x), \overline{\beta}(x)\right]$ be fixed. Let $\underline{\alpha}, \overline{\alpha} \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ be angles such that $\underline{\alpha} \leqslant \overline{\alpha}$; such that there exists a Type 1 admissible path in $\mathfrak{D}$ from $(W,\overline{\alpha})$ to (X,β); and such that there exists a Type 1 admissible path in $\mathfrak{D}$ from $(W,\underline{\alpha})$ to (X,β). Then, for every $\alpha \in [\underline{\alpha}, \overline{\alpha}]$, there exists a Type 1 admissible path in $\mathfrak{D}$ from (W,α) to (X,β).*

Proof. Let $\alpha \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$, and let $\Gamma_{\mathrm{i},0}$ (resp. $\Gamma_{\mathrm{f},1}$) be the C^+ (resp. C^-) arc passing through W (resp. X) with $\Gamma'_{\mathrm{i},0}(W) = \alpha$ (resp. $\Gamma'_{\mathrm{f},1}(X) = \beta$). If $\Gamma_{\mathrm{i},0}$ and $\Gamma_{\mathrm{f},1}$ do not intersect, then there exists a C^+SC^- path Π_α between (W,α) and (X,β). We define the function $\delta : \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \to [-\pi, \pi]$ as the angle of the slope of the S section of this path. It can be shown using elementary geometry that $\delta(\alpha)$ satisfies the equation

$$(x - w + r\cos\alpha + r\cos\beta)\cos(\delta(\alpha)) - (d_1 - r\sin\alpha - r\sin\beta)\sin(\delta(\alpha)) \;=\; 2r, \quad \text{(C.3)}$$

and $\Pi_\alpha = C^+_{r|\alpha-\delta(\alpha)|} S_\ell C^-_{r|\beta-\delta(\alpha)|}$, where

$$\ell := (x - w + r\cos\alpha + r\cos\beta - 2r\cos(\delta(\alpha))/\sin(\delta(\alpha)).$$

The path Π_α is Type 1 admissible if the points of tangency with $\Gamma_{\mathrm{i},0}$ and $\Gamma_{\mathrm{f},1}$ of the S section of Π_α lie in the rectangle $ABCD$, which in turn is true if $\alpha > \delta(\alpha)$, $\beta > \delta(\alpha)$, and $\alpha - \delta(\alpha) \leqslant \pi$, $\beta - \delta(\alpha) \leqslant \pi$.

We may now rewrite (C.3) as

$$A(\alpha)\cos\delta - B(\alpha)\sin\delta = 2r,$$

where $A(\alpha) := (x - w + r\cos\alpha + r\cos\beta)$, and $B(\alpha) := (d_1 - r\sin\alpha - r\sin\beta)$. It follows that

$$\begin{aligned} \frac{\partial\delta}{\partial\alpha}(\alpha) &= \frac{\left(\frac{\partial A}{\partial\alpha}(\alpha)\cos\delta(\alpha) - \frac{\partial B}{\partial\alpha}(\alpha)\sin\delta(\alpha)\right)}{A(\alpha)\sin\delta(\alpha) + B(\alpha)\cos\delta(\alpha)} && \text{(C.4)} \\ &= \frac{r\sin(\delta(\alpha) - \alpha)}{A(\alpha)\sin\delta(\alpha) + B(\alpha)\cos\delta(\alpha)}. && \text{(C.5)} \end{aligned}$$

It can be shown using elementary geometry that whenever there exists a Type 1 admissible C^+SC^- path between (W, α) and (X, β), $A(\alpha) > 0$ and $B(\alpha) > 0$. Then it follows from (C.3) that $\frac{A(\alpha)}{B(\alpha)} > \tan\delta$.

Now suppose, for the sake of contradiction, that $A(\alpha)\sin\delta + B(\alpha)\cos\delta \leqslant 0$. Then it follows that

$$\begin{aligned} & A(\alpha)\sin\delta(\alpha) \leqslant -B(\alpha)\cos\delta(\alpha), \\ \implies & \frac{-B(\alpha)}{A(\alpha)} \geqslant \tan\delta(\alpha), \\ \implies & \frac{-B(\alpha)}{A(\alpha)}\frac{A(\alpha)}{B(\alpha)} \geqslant \tan^2\delta(\alpha) \geqslant 0, \\ \implies & -1 \geqslant 0, \end{aligned}$$

which is a contradiction. Hence, $A(\alpha)\sin\delta(\alpha) + B(\alpha)\cos\delta(\alpha) > 0$. It follows from (C.4) that $\frac{\partial\delta}{\partial\alpha}(\alpha) \leqslant 0$ whenever $\delta(\alpha) \leqslant \alpha$, and $\delta(\alpha) - \alpha \geqslant -\pi$.

According to the hypothesis, there exists a Type 1 admissible path in $\mathfrak{D}$ between $(W, \overline{\alpha})$ and (X, β). Assume that this path is of type C^+SC^-. Then, the circles $\Gamma_{\mathrm{i},0}$ and $\Gamma_{\mathrm{f},1}$ do not intersect. It can be easily shown that $\Gamma_{\mathrm{i},0}$ and $\Gamma_{\mathrm{f},1}$ do not intersect for every $\alpha \in \left[-\frac{\pi}{2}, \overline{\alpha}\right]$. Also, by the earlier arguments, $\overline{\alpha} > \delta(\overline{\alpha})$, $\beta \geqslant \delta(\overline{\alpha})$, and $\frac{\partial\delta}{\partial\alpha}\big|_{\alpha=\overline{\alpha}} < 0$. Then it follows by the continuity of δ that there exists $\alpha_{SC^-} < \overline{\alpha}$ such

that $\delta(\alpha_{SC^-}) = \alpha_{SC^-}$. Geometrically, α_{SC^-} is the tangent angle at W of the SC^- path from W to X with tangent angle β at X. Furthermore, $\delta(\alpha_{SC^-}) < \beta$. Then it follows that for every $\alpha \in [\alpha_{SC^-}, \overline{\alpha}]$, there exists a C^+SC^- path Π_α between (W, α) and (X, β) such that the angle of slope of the S section of Π_α is $\delta(\alpha) \in [\delta(\overline{\alpha}), \alpha_{SC^-}]$. Since $\delta(\alpha) \leqslant \alpha$ for every $\alpha \in [\alpha_{SC^-}, \overline{\alpha}]$ and $\delta(\alpha) < \delta(\alpha_{SC^-}) < \beta$, it follows that Π_α is Type 1 admissible.

Using similar arguments, we can prove an identical statement in the case that the Type 1 admissible path in $\mathfrak{D}$ between $(W, \overline{\alpha})$ and (X, β) (guaranteed by the hypothesis) is a C^+SC^+ path. Repeating the same analyses after a reflection about the x-axis, we can summarize the results as follows:

(R1) If there exists a Type 1 admissible C^+SC^- path between $(W, \overline{\alpha})$ and (X, β), then for every $\alpha \in [\alpha_{SC^-}, \overline{\alpha}]$, there exists a Type 1 admissible C^+SC^- path between (W, α) and (X, β).

(R2) If there exists a Type 1 admissible C^+SC^+ path between $(W, \overline{\alpha})$ and (X, β), then for every $\alpha \in [\alpha_{SC^+}, \overline{\alpha}]$, there exists a Type 1 admissible C^+SC^+ path between (W, α) and (X, β).

(R3) If there exists a Type 1 admissible C^-SC^- path between $(W, \underline{\alpha})$ and (X, β), then for every $\alpha \in [\underline{\alpha}, \alpha_{SC^-}]$, there exists a Type 1 admissible C^-SC^- path between (W, α) and (X, β).

(R4) If there exists a Type 1 admissible C^-SC^+ path between $(W, \underline{\alpha})$ and (X, β), then for every $\alpha \in [\underline{\alpha}, \alpha_{SC^+}]$, there exists a Type 1 admissible C^-SC^+ path between (W, α) and (X, β).

By the hypothesis, there exist two paths in $\Pi, \bar{\Pi} \in \mathfrak{D}$ such that $\Pi'(W) = \overline{\alpha}$ and $\bar{\Pi}'(W) = \underline{\alpha}$. By Facts C.5 and C.6, it follows that either of the following cases holds true:

1. Both Π and $\bar{\Pi}$ are of the same type (i.e., C^+SC^-, C^+SC^+, C^-SC^-, or C^-SC^+)

2. Π is of type C^+SC^- and $\bar{\Pi}$ is of type C^-SC^-

3. Π is of type C^+SC^+ and $\bar{\Pi}$ is of type C^-SC^+

In the first case, exactly one of the four results (R1),...,(R4) suffices to prove the Lemma. In the second case, the combination of (R1) and (R3) proves the Lemma. Finally, in the third case, the combination of (R2) and (R4) proves the Lemma. □

Lemma C.8. *Let X be a point on the segment YZ. If there exists a Type 1 admissible C^+ path $\Gamma_{i,0}$ between W and X, then the tangent angle at W of any Type 1 admissible path between W and X is no greater than $\Gamma'_{i,0}(W)$.*

Proof. The absolute rate of change of orientation along a continuously differentiable path is by definition the curvature of the path. In the family of paths with curvature at most r^{-1} everywhere, a circle of radius r is the path where the absolute rate of change of orientation is maximum, since the curvature along the circle is equal to r^{-1} everywhere. It follows that any continuously differentiable path Π from W to Q with $\Pi'(W) > \Gamma'_{i,0}(W)$ and $\Pi'(X) \leqslant 0$ necessarily has curvature greater than r^{-1} at some point on that curve.

Now suppose, for the sake of contradiction, that there exists a Type 1 admissible path Π between W and X which satisfies $\Pi'(W) > \Gamma'_{i,0}(W)$ and $\Pi'(X) > \Gamma'_{i,0}(X)$. By continuity of the tangent, Π intersects $\Gamma_{i,0}$ at a point P between W and X. Then $\Pi'(W) > \Gamma'_{i,0}(W)$ (by assumption) and $\Pi'(P) < \Gamma'_{i,0}(P)$, which implies that the curvature of the section of Π between W and P exceeds r^{-1} at some point on that section of Π. This contradicts the assumption that the curvature at all points on Π is at most r^{-1}. □

Corollary C.9. *If there exists a Type 1 admissible C^- path $\Gamma_{i,1}$ between W and X, then the tangent angle at W of any Type 1 admissible path between W and X is no less than $\Gamma'_{i,1}(W)$.*

Corollary C.10. *If there exists a Type 1 admissible C^+ path $\Gamma_{\mathrm{i},0}$ between W and X, then the tangent angle at X of any Type 1 admissible path between W and X is no less than $\Gamma'_{\mathrm{i},0}(X)$.*

Corollary C.11. *If there exists a Type 1 admissible C^- path $\Gamma_{\mathrm{i},1}$ between W and X, then the tangent angle at X of any Type 1 admissible path between W and X is no greater than $\Gamma'_{\mathrm{i},1}(X)$.*

Corollaries C.9 and C.10 are direct consequences of Lemma C.8 after reflections about the horizontal and vertical axes respectively. Corollary C.11 is a consequence of Corollary C.10 after a reflection about the horizontal axes.

We are now prepared to present the constructions of Υ_x and Λ_x for the traversal of parallel edges. For the ease of analysis, we identify the following mutually exclusive cases.

Case 1: $0 < r < (d_1/4)$,

Case 2: $(d_1/4) \leqslant r < (d_1/2)$,

Case 3: $(d_1/2) \leqslant r$.

C.1.1 Case 1

Let $X = (d_1, x)$ be a point on the segment YZ, and let $\beta \in [\underline{\alpha}(x), \overline{\alpha}(x)]$. Let $\Gamma_{\mathrm{i},0}$ be the C^+ arc passing through W such that $\Gamma'_{\mathrm{i},0}(W) = \alpha^*(w)$. Let $\Gamma_{\mathrm{f},0}$ and $\Gamma_{\mathrm{f},1}$ be, respectively, the C^+ and C^- arcs passing through X such that $\Gamma'_{\mathrm{f},0}(X) = \Gamma'_{\mathrm{f},1}(X) = \beta$. The arc $\Gamma_{\mathrm{i},0}$ does not intersect either of the arcs $\Gamma_{\mathrm{f},0}$ and $\Gamma_{\mathrm{f},1}$, since $r < d_1/4$. Hence, there exists a C^+SC^- path as well as a C^+SC^+ path from $(W, \alpha^*(w))$ to (X, β), but we need to show that one of these paths is Type 1 admissible.

For every $\gamma \in (\beta, \frac{\pi}{2}]$, let $\Psi_1(\gamma)$ be the line with angle of slope γ that is tangent to $\Gamma_{\mathrm{f},0}$. Similarly, for every $\delta \in \left[-\frac{\pi}{2}, \beta\right)$, let $\Psi_2(\delta)$ be the line with angle of slope δ that

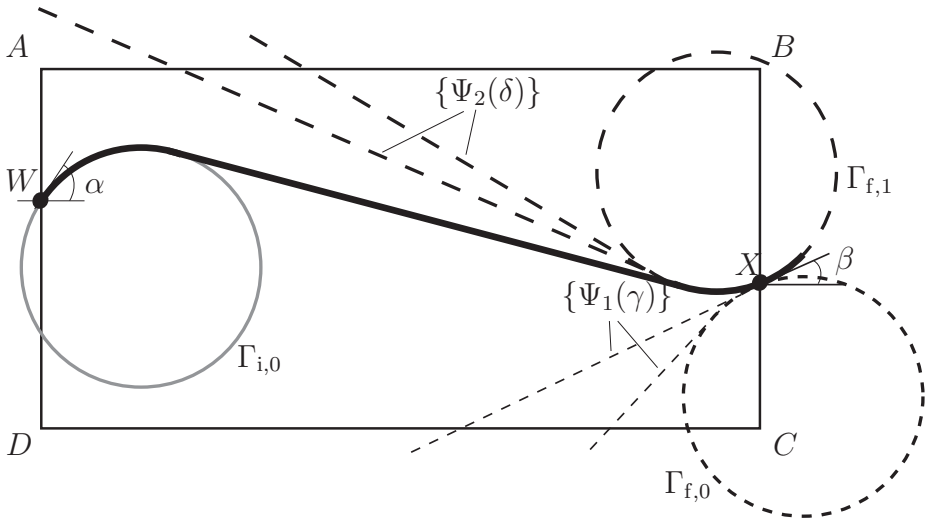


Figure C.2: Construction of Υ_x for Case 1.

is tangent to $\Gamma_{\mathrm{f},1}$. Finally, let Ψ_3 be the line passing through the point X with angle of slope β.

When traversing a C^+ arc (resp. C^- arc), the orientation along the path decreases monotonically (resp. increases monotonically). Hence, for each $\gamma \in (\beta, \frac{\pi}{2}]$ (resp. $\delta \in \left[-\frac{\pi}{2}, \beta\right)$), the point of tangency of $\Psi_1(\gamma)$ to $\Gamma_{\mathrm{f},0}$ (resp. $\Psi_2(\delta)$ to $\Gamma_{\mathrm{f},1}$) lies in the interior of the rectangle $ABCD$. Also, note that the orientation along the arc $\Gamma_{\mathrm{i},0}$ decreases monotonically starting from $\alpha^*(w)$. It follows that there exists $\eta \in \left[-\frac{\pi}{2}, \alpha^*(w)\right]$ such that the line Ψ with slope $\tan\eta$ which is tangent to $\Gamma_{\mathrm{i},0}$ belongs to the set

$$\left\{\Psi_1(\gamma)\right\}_{\gamma\in(\beta,\frac{\pi}{2}]} \bigcup \left\{\Psi_2(\delta)\right\}_{\delta\in\left[-\frac{\pi}{2},\beta\right)} \bigcup \left\{\Psi_3\right\}.$$

Thus, the line Ψ is either tangent to $\Gamma_{\mathrm{f},0}$ or to $\Gamma_{\mathrm{f},1}$ at a point in the interior of the rectangle $ABCD$, or it is tangent to both $\Gamma_{\mathrm{f},0}$ and $\Gamma_{\mathrm{f},1}$ at the point X. Hence, the path Π_x defined by

$$\Pi_x := \begin{cases} C^+_{|\alpha^*(w)-\eta|} SC^+_{|\eta-\beta|}, & \Psi \in \left\{\Psi_1(\gamma)\right\}_{\gamma\in(\beta,\frac{\pi}{2}]}, \\ C^+_{|\alpha^*(w)-\eta|} SC^-_{|\eta-\beta|}, & \Psi \in \left\{\Psi_2(\delta)\right\}_{\delta\in\left[-\frac{\pi}{2},\beta\right)}, \\ C^+_{|\alpha^*(w)-\eta|} S & \Psi = \Psi_3, \end{cases} \tag{C.6}$$

is a Type 1 admissible path from W to X. The family of paths Π_x is our candidate

Computation of $\Upsilon'_x(W)$ and $\overline{\alpha}$ for Case 1 (Parallel Edges)

1: $\Upsilon'_x(W) \leftarrow \alpha^*(w)$, for all $x \in [y, z]$
2: $\mathcal{I} \leftarrow [y, z]$
3: $\overline{\alpha} \leftarrow \max_{x \in \mathcal{I}} \{\Upsilon'_x(W)\} = \alpha^*(w)$

Figure C.3: Analysis of traversal across parallel edges: $0 < r < (d_1/4)$

for the family Υ_x. As a candidate for the family Λ_x, we construct the family of paths $\bar{\Pi}_x$ by following an identical procedure as above after a reflection about the x-axis, i.e., $\bar{\Pi}_x$ is either a C^-SC^+, or C^-SC^-, or a C^-S path from W to X with $\bar{\Pi}'_x(W) = -\alpha^*(d_2 - w)$.

To show that $\Pi_x = \Upsilon_x$ and $\bar{\Pi}_x = \Lambda_x$, we must show that the families Π_x and $\bar{\Pi}_x$ satisfy the properties (P1) through (P5) listed in Section 5.2.

Proof of (P1). Since our choice of X in the construction described above was arbitrary, $\Pi'_x(W) = \alpha^*(w)$ and $\bar{\Pi}'_x(W) = -\alpha^*(d_2 - w)$, for every $x \in [y, z]$. It follows that (P1), namely, $\Pi'_x(w) \leqslant \bar{\Pi}'_x(w)$, holds true.

Proof of (P2). By construction, the paths Π_x and $\bar{\Pi}_x$ belong to $\mathfrak{D}$. Thus, (P2) follows as a direct consequence of Lemma C.7.

Proofs of (P4) and (P5). We note that

$$\begin{aligned} \max_{y \leqslant x \leqslant z} \{\Pi'_x(W)\} &= \max_{y \leqslant x \leqslant z} \alpha^*(w) = \alpha^*(w), \\ \min_{y \leqslant x \leqslant z} \{\bar{\Pi}'_x(W)\} &= \min_{y \leqslant x \leqslant z} -\alpha^*(d_2 - w) = -\alpha^*(d_2 - w). \end{aligned}$$

Since $r \leqslant \dfrac{d_1^2 + (d_2 - w)^2}{2(d_2 - w)}$ for Case 1, it follows by Lemma C.2 that for every $x \in [y, z]$, there exists no Type 1 admissible path Φ_x such that $\Phi'_x(W) > \Pi'_x(W)$. Similarly, applying Lemma C.2 after a reflection about the x-axis, it follows that for every $x \in [y, z]$, there exists no Type 1 admissible path Φ_x such that $\Phi'_x(W) < \bar{\Pi}'_x(W)$.

Proof of (P3). As noted above, the angles $\Pi'_x(W) = \alpha^*(w)$ and $\bar{\Pi}'_x(W) = -\alpha^*(d_2 - w)$ for every $x \in [y, z]$. In particular, $\Pi'_x(W)$ and $\bar{\Pi}'_x(W)$ are continuous with respect to x.

We have thus established that $\Pi = \Upsilon_x$ and $\bar{\Pi} = \Lambda_x$, and consequently, the computation of $\overline{\alpha}$ for Case 1 may be performed as described in Fig. C.3.

C.1.2 Case 2

The computation of $\overline{\alpha}$ is described in Fig. C.4; in what follows, we discuss the constructions of paths underlying the procedure in Fig. C.4.

First, suppose $3r + r\sin\alpha^*(w) < d_1$. Let $\Gamma_{\mathrm{i},0}$ be the C^+ arc passing through W with $\Gamma'_{\mathrm{i},0}(W) = \alpha^*(w)$ and let $\Gamma_{\mathrm{f},0}$ and $\Gamma_{\mathrm{f},1}$ be, respectively, the C^+ and C^- arcs passing through X with $\Gamma'_{\mathrm{f},0}(X) = \Gamma'_{\mathrm{f},1}(X) = \beta$. It can be shown using elementary geometry that $\Gamma_{\mathrm{i},0}$ does not intersect with either $\Gamma_{\mathrm{f},0}$ or with $\Gamma_{\mathrm{f},1}$ (see Fig C.5(a)). It then follows using the arguments in Section C.1.1 that there exists a C^+SC^+ path a C^+SC^- path from $(W, \alpha^*(w))$ to (X, β) for every point X on the segment YZ and for every $\beta \in \left[\underline{\beta}(x), \overline{\beta}(x)\right]$. This observation is the basis for Line 1-2.

Now suppose that $3r + r\sin\alpha^*(w) < d_1$, and let $\Gamma_{\mathrm{i},0}$ be as before. Let $\Gamma_{\mathrm{f},1}$ be the C^- arc that is tangent to both $\Gamma_{\mathrm{i},0}$ and the line BC, such that the center of $\Gamma_{\mathrm{f},1}$ lies above the center of $\Gamma_{\mathrm{i},0}$ (see Fig. C.5(b)). It can be shown using elementary geometry that the ordinate of the point $M_2 = (d_1, m_2)$ of tangency of $\Gamma_{\mathrm{f},1}$ with line BC is given by (5). Also, let $M_1 = (d_1, m_1)$ be the lower point of intersection of line BC with the circle that is concentric with $\Gamma_{\mathrm{i},0}$ and has radius $3r$ (see Fig C.5(b)). To construct candidate paths for the family Υ_x, we consider three sub-cases, as follows.

A. Let $X = (d_1, x)$ be a point on the segment YZ such that $x \in [y, m_1)$ if $m_1 \geqslant y$ (otherwise we ignore this case). For $\beta \in [\underline{\alpha}(x), \overline{\alpha}(x)]$, let $\Gamma_{\mathrm{f},0}$ and $\Gamma_{\mathrm{f},1}$ be, respectively, the C^+ and C^- arcs passing through X with $\Gamma'_{\mathrm{f},0}(X) = \Gamma'_{\mathrm{f},1}(X) = \beta$. It follows from the definition of the point M_1 that $\Gamma_{\mathrm{f},0}$ and $\Gamma_{\mathrm{f},1}$ do not intersect

Computation of $\Upsilon'_x(W)$ and $\overline{\alpha}$ for Case 2 (Parallel Edges)

if $3r + r\sin\alpha^*(w) < d_1$ **then**

  $\Upsilon'_x(W) \leftarrow \alpha^*(w)$, for all $x \in [y, z]$

else

  $m_1 \leftarrow w - \sqrt{4r^2 - (r\sin(\alpha^*(w)) - d_1)^2} - r\cos(\alpha^*(w))$

  $m_2 \leftarrow w + \sqrt{4r^2 - (r\sin(\alpha^*(w)) - (d_1 - r))^2} - r\cos(\alpha^*(w))$

  if $m_1 \geqslant y$ or $m_2 \leqslant z$ **then**

    $\Upsilon'_x(W) \leftarrow \alpha^*(w)$, for all $x \in [y, m_1] \cup [m_2, z]$

  Compute $\beta^*(x)$, for all $x \in (m_1, m_2) \cap [y, z]$, as the solution in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ to

$$\left(\cos\alpha^*(w) + \frac{x-w}{r}\right)\cos(\beta^*(x)) + \left(\sin\alpha^*(w) - \frac{d_1}{r}\right)\sin(\beta^*(x)) + \frac{x-w}{r}\cos\alpha^*(w) - \frac{d_1}{r}\sin\alpha^*(w) + \frac{(x-w)^2 + d_1^2}{2r^2} = 1 \tag{C.7}$$

  if $\beta^*(x) < \underline{\beta}(x)$ **then**

    $\Upsilon'_x(W)$ is the solution in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ to the equation

$$\left(\cos\underline{\beta}(x) + \frac{x-w}{r}\right)\cos(\Upsilon'_x(W)) + \left(\sin\underline{\beta}(x) - \frac{d_1}{r}\right)\sin(\Upsilon'_x(W)) + \frac{x-w}{r}\cos\underline{\beta}(x) - \frac{d_1}{r}\sin\underline{\beta}(x) + \frac{(x-w)^2 + d_1^2}{2r^2} = 1 \tag{C.8}$$

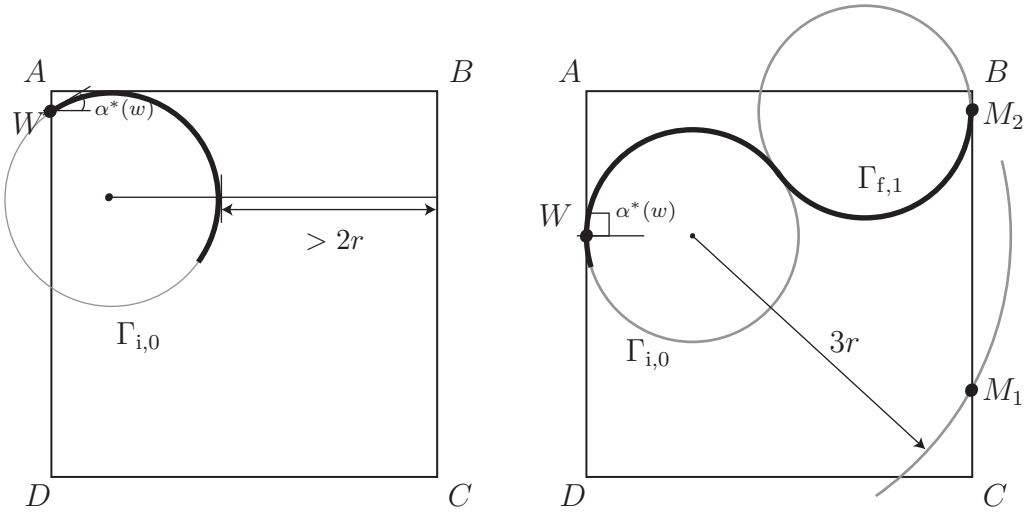
  else

    $\Upsilon'_x(W) \leftarrow \alpha^*(w)$

$\mathcal{I} \leftarrow [y, z]$

$\overline{\alpha} \leftarrow \max_{x\in\mathcal{I}}\{\Upsilon'_x(W)\}$

Figure C.4: Analysis of traversal across parallel edges: $(d_1/4) < r < (d_1/2)$.



(a) Case 2, $3r + r\sin\alpha^*(w) < d_1$. (b) Locations of M_1 and M_2.

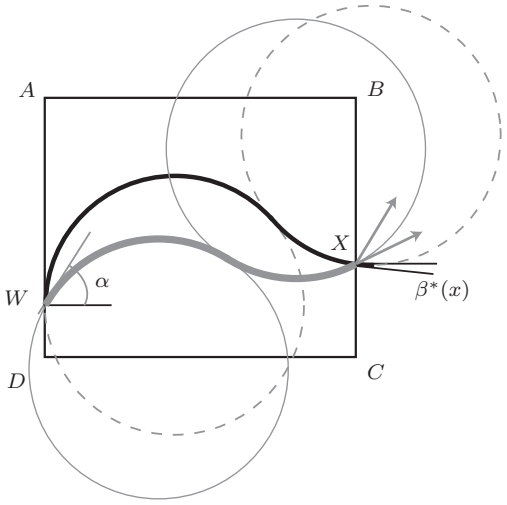
Figure C.5: Analysis of Case 2.

with $\Gamma_{\mathrm{i},0}$. Then, using arguments described in Section C.1.1, we may construct a Type 1 admissible C^+SC^+ or C^+SC^- path Π_x between $(W, \alpha^*(w))$ and (X, β).

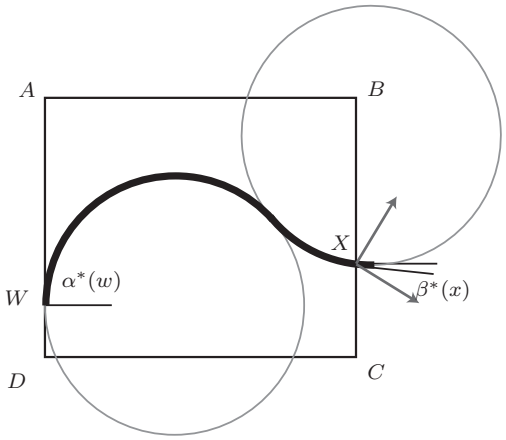
B. Let $X = (d_1, x)$ be a point on the segment YZ such that $x \in [m_1, m_2] \cap [y, z]$. Let Π_x be the C^+C^- path from W to X with $\Pi'_x(W) = \alpha^*(w)$, and define $\beta^*(x) := \Pi'_x(X)$; it can be shown that $\beta^*(x)$ satisfies (C.7).

If $\beta^*(x) < \underline{\beta}(x)$, then Π_x is not Type 1 admissible (see Fig. C.6(a)), and moreover, there exists no Type 1 admissible path from W to X with tangent angle $\alpha^*(w)$ at W. Therefore, we redefine Π_x as the C^+C^- path from W to X which satisfies $\Pi'_x(X) = \underline{\beta}(x)$ (see Fig. C.6(c)). This path Π_x is Type 1 admissible, and $\Pi'_x(W)$ satisfies (C.8).

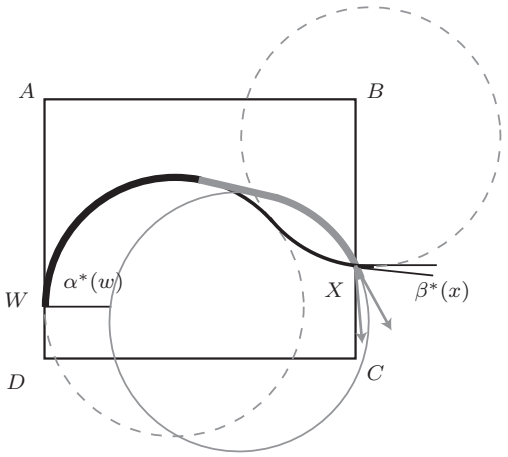
If $\underline{\beta}(x) \leqslant \beta^*(x) \leqslant \overline{\beta}(x)$, then Π_x is Type 1 admissible (see Fig. C.6(b)). Finally, if $\overline{\beta}(x) < \beta^*(x)$, then Π_x is not Type 1 admissible. Let $\Gamma_{\mathrm{i},0}$ be the C^+ arc passing through W with $\Gamma'_{\mathrm{i},0}(W) = \alpha^*(w)$ and let $\Gamma_{\mathrm{f},0}$ be the C^+ arc passing through X with $\Gamma'_{\mathrm{f},0}(X) = \overline{\beta}(x)$. We redefine Π_x as the C^+SC^+ path consisting of sections of $\Gamma_{\mathrm{i},0}$ and $\Gamma_{\mathrm{f},0}$, and it can be shown that Π_x is Type 1 admissible.



(a) $\beta^*(x) < \underline{\beta}(x)$, originally defined Π_x (the path in black) is not admissible, redefined Π_x shown in gray.



(b) $\underline{\beta}(x) \leqslant \beta^*(x) \leqslant \overline{\beta}(x)$, Π_x is admissible.



(c) $\overline{\beta}(x) < \beta^*(x)$, originally defined Π_x (the path in black) is not admissible, Π_x redefined as a C^+SC^+ path (shown in gray).

Figure C.6: Constructions of Υ_x for Case 2, sub-case B. The arrows on segment BC indicate terminal orientation cones.

C. Let $X = (d_1, x)$ be a point on the segment YZ such that $x \in (m_2, z]$ if $m_2 \leqslant z$ (otherwise we ignore this case). Let $\Gamma_{\mathrm{i},0}$ be the C^+ arc passing through W with $\Gamma'_{\mathrm{i},0}(W) = \alpha^*(w)$. Let δ be the angle of slope of the upper tangent line to $\Gamma_{\mathrm{i},0}$ that passes through X, let $\beta \in (\delta, \frac{\pi}{2}] \cap [\underline{\alpha}(x), \overline{\alpha}(x)]$, and let $\Gamma_{\mathrm{f},1}$ be the C^- arc passing through X with $\Gamma'_{\mathrm{f},1}(X) = \beta$. It follows by elementary geometry that $\Gamma_{\mathrm{i},0}$ and $\Gamma_{\mathrm{f},1}$ do not intersect. Then, using arguments described in Section C.1.1, we may construct a Type 1 admissible C^+SC^- path Π_x between $(W, \alpha^*(w))$ and (X, β) consisting of sections of $\Gamma_{\mathrm{i},0}$ and $\Gamma_{\mathrm{f},1}$.

In summary, we can construct a family of Type 1 admissible paths Π_x such that Π_x is either a C^+SC^+, C^+SC^-, or a C^+C^- path, i.e., $\Pi_x \in \mathfrak{D}$. Using identical arguments after a reflection about the x-axis, we can construct a family of paths $\bar{\Pi}_x$ such that $\bar{\Pi}_x \in \mathfrak{D}$. To show that $\Pi_x = \Upsilon_x$ and $\bar{\Pi}_x = \Lambda_x$, we show that the families Π_x and $\bar{\Pi}_x$ satisfy the properties (P1)–(P5).

Proof of (P1). Note that in the above constructions, $\Pi'_x(W) = \alpha^*(w)$ for every $x \in [y, z]$, except when $\beta^*(x) < \underline{\beta}(x)$, in sub-case B. Similarly, $\bar{\Pi}'_x(W) = -\alpha^*(d_2 - w)$ for every $x \in [y, z]$, except when $\beta_*(x) > \overline{\beta}(x)$, where $\beta_*(x)$ is the tangent angle at X of the C^-C^+ path between W and X which has tangent angle $-\alpha^*(d_2 - w)$ at W. It can be shown using elementary geometry that $\beta_*(x) < \beta^*(x)$ for every $x \in [y, z]$, and it follows that the conditions $\beta^*(x) < \underline{\beta}(x)$ and $\beta_*(x) > \overline{\beta}(x)$ never occur *simultaneously* for any $x \in [y, z]$, since $\underline{\beta}(x) \leqslant \overline{\beta}(x)$, for every $x \in [y, z]$. Thus, for every $x \in [y, z]$, either $\Pi'_x(W) = \alpha^*(w)$ or $\bar{\Pi}'_x(W) = -\alpha^*(d_2 - w)$ or both, and it follows by Lemma C.2 that (P1) holds true.

Proof of (P2). By construction, the paths Π_x and $\bar{\Pi}_x$ belong to $\mathfrak{D}$. Thus, property (P2) follows as a direct consequence of Lemma C.7.

Proof of (P3). If $m_1 \geqslant y$, and/or $m_2 \leqslant z$, then $\Upsilon'_x(W) = \alpha^*(w)$, for $x \in [y, m_1) \cup (m_2, z]$ and hence $\Upsilon'_x(W)$ is continuous on the set $[y, m_1) \cup (m_2, z]$. For $x \in [m_1, m_2]$, $\Upsilon'_x(W)$ is either $\alpha^*(w)$ or it is the solution to (C.8), depending on the value of $\beta^*(x)$. It follows that $\Upsilon'_x(W)$ is continuous wherever $\beta^*(x) < \underline{\beta}(x)$, and by (C.8), it is continuous wherever $\beta^*(x) \geqslant \underline{\beta}(x)$. By (C.7), β^* is continuous. Since $\underline{\beta}$ is also continuous, to show that $\Upsilon'_x(W)$ is continuous over the entire interval $[m_1, m_2]$, it suffices to show that when $\beta^*(x) = \underline{\beta}(x)$, $\alpha^*(w)$ is a solution to (C.8). To see this, substitute $\Upsilon'_x(W) = \alpha^*(w)$ and $\beta^*(x) = \underline{\beta}(x)$ in (C.8) and then subtract (C.7) from the resulting equation, to arrive at the identity $0 = 0$. Finally, to show that $\Upsilon'_x(W)$ is continuous on $[y, z]$, it suffices to show that $\Upsilon'_{m_1}(W) = \Upsilon'_{m_2}(W) = \alpha^*(w)$. To see this, note that the C^- arc passing through M_1 with tangent angle $\beta^*(m_1)$ is tangent to the C^+ arc $\Gamma_{\text{i},0}$ passing through W with tangent angle $\alpha^*(w)$ at W, and no other C^- arc passing through M_1 with tangent angle at M_1 in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ intersects $\Gamma_{\text{i},0}$. Thus, there exists a Type 1 admissible C^+C^- or C^+SC^- path from W to M_1 consisting $\Gamma_{\text{f},0}$ as the first arc. Thus $\Upsilon'_{m_1}(W) = \alpha^*(w)$. Similarly, the C^- arc passing through M_2 with tangent angle $\frac{\pi}{2}$ at M_2 is tangent to $\Gamma_{\text{i},0}$, and no other C^- arc through M_2 with tangent angle in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ intersects $\Gamma_{\text{i},0}$, and it follows that $\Upsilon'_{m_2}(W) = \alpha^*(w)$.

Proofs of (P4) and (P5). As noted above, $\Pi'_x(W) = \alpha^*(w)$ for every $x \in [y, z]$, except when $\beta^*(x) < \underline{\beta}(x)$, in sub-case B. Thus, we only need to show that (P4) is satisfied when $\beta^*(x) < \underline{\beta}(x)$, and Π_x is redefined as the C^+C^- path from W to X such that $\Pi'_x(X) = \underline{\beta}(x)$.

Let P be a point in the interior of the rectangle $ABCD$ and let S and T be, respectively, the points of intersection of the line passing through P that is parallel to the y-axis with lines AB and CD, and consider the problem of traversal across parallel edges in rectangle $ASTC$. It follows from Lemma C.8 that if there exists a C^+ path $\Gamma_{\text{i},0}$ between W and P, then the tangent angle at W of any Type 1 admissible

path[2] (in rectangle $ASTC$) is no greater than $\Gamma'_{\mathrm{i},0}(W)$.

Moreover, it follows by Corollaries C.9 and C.11 that if there exists a C^- path $\Gamma_{\mathrm{f},1}$ between P and X, then there exists no Type 1 admissible path (in rectangle $SBCT$) with tangent angle less than $\Gamma'_{\mathrm{f},1}(P)$ at P or with tangent angle greater than $\Gamma'_{\mathrm{f},1}(X)$ at X. However, to satisfy the terminal orientation cone condition, we require that the $\underline{\beta}(x) \leqslant \Gamma'_{\mathrm{f},1}(X) \leqslant \overline{\beta}(x)$. The concatenation of a Type 1 admissible path in rectangle $ASTC$ with a Type 1 admissible path in rectangle $SBCT$ is a Type 1 admissible path in the rectangle $ABCD$. It follows that for any $\beta \in \left[\underline{\beta}(x), \overline{\beta}(x)\right]$, if there exists a point P in the interior of the rectangle $ABCD$ such that a Type 1 admissible C^+C^- path $\Pi_{x,\beta}$ can be constructed as above between (W, α) and (X, β), then there exists no other Type 1 admissible path between these configurations with tangent angle at W greater than $\Pi'_{x,\beta}(W)$. The analysis of (C.8) shows that $\Pi'_{x,\beta}(W)$ decreases monotonically with increasing β for $\beta \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$, and it follows that Π_x as defined earlier satisfies (P4). We may use identical arguments after a reflection about the x-axis to show that $\bar{\Pi}_x$ satisfies (P5).

C.1.3 Case 3

The computation of $\overline{\alpha}$ is described in Fig. C.4. Note that the procedure for computing $\Upsilon'_x(W)$ provided in Chapter 5 corresponds to the steps in Fig. C.7. In what follows, we discuss the constructions of paths underlying the procedure in Fig. C.4. First, suppose

$$r \leqslant (d_1^2 + (d_2 - w)^2)/2(d_2 - w) \quad \text{and} \quad r + \sqrt{2r(d_2 - w) - (d_2 - w)^2} < d_1.$$

It can be shown using elementary geometry that, for every $\alpha \in \left[-\frac{\pi}{2}, \alpha^*(w)\right]$, the C^+ arc $\Gamma_{\mathrm{i},0}$ passing through W with $\Gamma'_{\mathrm{i},0}(W) = \alpha$ does not intersect the segment BC, and the entire analysis of Case 2 applies (see Fig. C.8(a)).

[2]For the rectangle $ASTC$, we consider the terminal orientation cone to be free, i.e., the tangent angle of admissible paths at P can be any angle in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$.

Computation of $\Upsilon'_x(W)$ and $\overline{\alpha}$ for Case 3 (Parallel Edges)

1: **if** $r \leqslant \dfrac{(d_1^2 + (d_2 - w)^2)}{2(d_2 - w)}$ and $r + \sqrt{2r(d_2 - w) - (d_2 - w)^2} < d_1$ **then**
2: Execute the procedure in Fig. C.4
3: **else**
4: Define m_1 and m_2 as in Lines 4–5 of Fig. C.4
5: $n_1 \leftarrow w - \sqrt{2rd_1 - d_1^2}$
6: $n_2 \leftarrow w + \sqrt{r^2 - \left(r\sin\left(\alpha^*(w)\right) - d_1\right)^2} - r\cos\left(\alpha^*(w)\right)$
7: **if** $m_1 \geqslant y$ or $m_2 \leqslant z$ **then**
8: $\Upsilon'_x(W) \leftarrow \alpha^*(w), \quad$ for all $x \in [y, m_1] \cup [m_2, z]$
9: **if** $n_1 \geqslant y$ or $n_2 \leqslant z$ **then**
10: For all $x \in \{(m_1, n_2) \cup (n_2, m_2)\} \cap [y, z]$ execute Lines 8–14 of Fig. C.4
11: Compute $\gamma^*(x)$ for all $x \in [n_1, n_2] \cap [y, z]$ as the solution in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ to

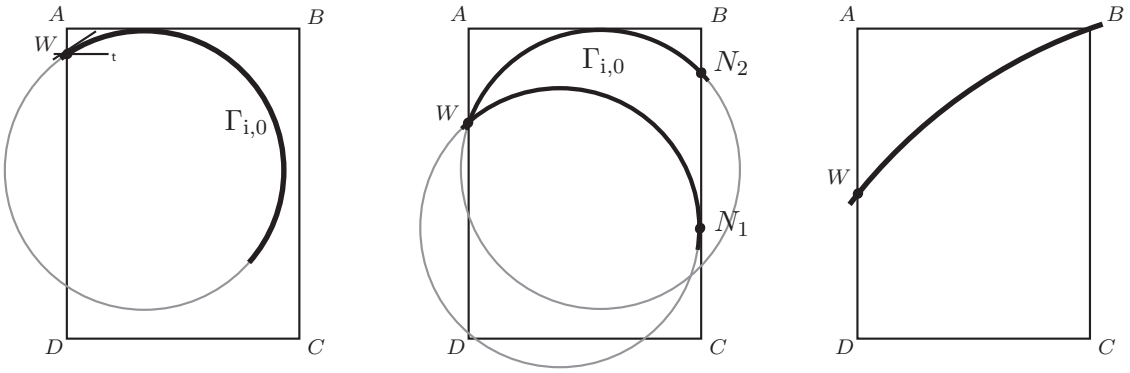
$$(x - w)\cos\left(\gamma^*(x)\right) - d_1\sin\left(\gamma^*(x)\right) \;=\; \frac{d_1^2 + (x - w)^2}{2r} \qquad \text{(C.9)}$$

12: **if** $\gamma^*(x) < \underline{\beta}(x)$ **then**
13: $\Upsilon'_x(W)$ is the solution in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ to (C.8)
14: **if** $\underline{\beta}(x) \leqslant \gamma^*(x) < \overline{\beta}(x)$ **then**
15: $\Upsilon'_x(W)$ is the solution in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ to the equation

$$-(x - w)\cos\left(\Upsilon'_x(W)\right) + d_1\sin\left(\Upsilon'_x(W)\right) \;=\; \frac{d_1^2 + (x - w)^2}{2r} \qquad \text{(C.10)}$$

16: **else**
17: There exists no Type 1 admissible path between W and X, in particular, Υ_x does not exist
18: $\mathcal{I} \leftarrow$ largest interval in the set $[y, z] \setminus \left\{x \in \left([n_1, n_2] \cap [y, z]\right) : \overline{\beta}(x) < \gamma^*(x)\right\}$
19: $\overline{\alpha} \leftarrow \max_{x \in \mathcal{I}} \left\{\Upsilon'_x(W)\right\}$

Figure C.7: Analysis of traversal across parallel edges: $(d_1/2) < r$.



(a) $\dfrac{r}{\sqrt{2r(d_2-w)-(d_2-w)^2}} + r < d_1$.

(b) $\dfrac{r}{\sqrt{2r(d_2-w)-(d_2-w)^2}} + r \geqslant d_1$ and $r \leqslant \dfrac{d_1^2+(d_2-w)^2}{2(d_2-w)}$.

(c) $r > \dfrac{d_1^2+(d_2-w)^2}{2(d_2-w)}$.

Figure C.8: Analysis for Case 3.

Now suppose that either

$$r \leqslant (d_1^2+(d_2-w)^2)/2(d_2-w) \quad \text{and} \quad r+\sqrt{2r(d_2-w)-(d_2-w)^2} \geqslant d_1$$

holds true, or $r > (d_1^2+(d_2-w)^2)/2(d_2-w)$ holds true. Consider the circle which passes through W and is tangent to the the line BC such that the point of tangency $N_1 = (d_1, n_1)$ has ordinate less than w (see Fig. C.8(b)). It can be shown that n_1 is given by (5). If $r \leqslant (d_1^2+(d_2-w)^2)/(2(d_2-w))$, then let $N_2 = (d_1, n_2)$ be the point of intersection with line BC of the C^+ arc $\Gamma_{i,0}$ that passes through W with $\Gamma'_{i,0}(W) = \alpha^*(w)$. It can be shown that n_2 is given by (6). Otherwise, if $r > (d_1^2+(d_2-w)^2)/(2(d_2-w))$, then $N_2 := Z$ (see Fig. C.8(c)). In addition, let M_1 and M_2 be points as defined in Section C.1.2 if $r \leqslant (d_1^2+(d_2-w)^2)/(2(d_2-w))$. Otherwise, if $r > (d_1^2+(d_2-w)^2)/(2(d_2-w))$, M_1 is as before and $M_2 := Z$. To construct candidate paths for the family Υ_x, we consider five sub-cases, as follows.

A. Let $X = (d_1, x)$ be a point on the segment YZ such that $x \in [y, m_1)$ if $m_1 \geqslant y$ (otherwise we ignore this case). This sub-case is identical to sub-case A of Case 2.

B. Let $X = (d_1, x)$ be a point on the segment YZ such that $x \in [m_1, n_1) \cap [y, z]$ if

$n_1 \geqslant y$ (otherwise we ignore this case). This sub-case is identical to sub-case B of Case 2.

C. Let $X = (d_1, x)$ be a point on the segment YZ such that $x \in [n_1, n_2] \cap [y, z]$. By definition of the points N_1 and N_2, there exists a C^+ path Π_x between W and X. We define the function $x \mapsto \gamma^*(x)$ as $\gamma^*(x) := \Pi'_x(X)$, and it can be shown using elementary geometry that $\gamma^*(x)$ satisfies (C.9).

If $\gamma^*(x) < \underline{\beta}(x)$, then Π_x is not Type 1 admissible (see Fig C.9(a)). We redefine Π_x as the C^+C^- path from W to X which satisfies $\Pi'_x(X) = \underline{\beta}(x)$. This path Π_x is Type 1 admissible, and $\Pi'_x(W)$ satisfies (C.8).

If $\underline{\beta}(x) \leqslant \gamma^*(x) < \overline{\beta}(x)$, then Π_x is Type 1 admissible (see Fig C.9(b)), and $\Pi'_x(W)$ satisfies (C.10).

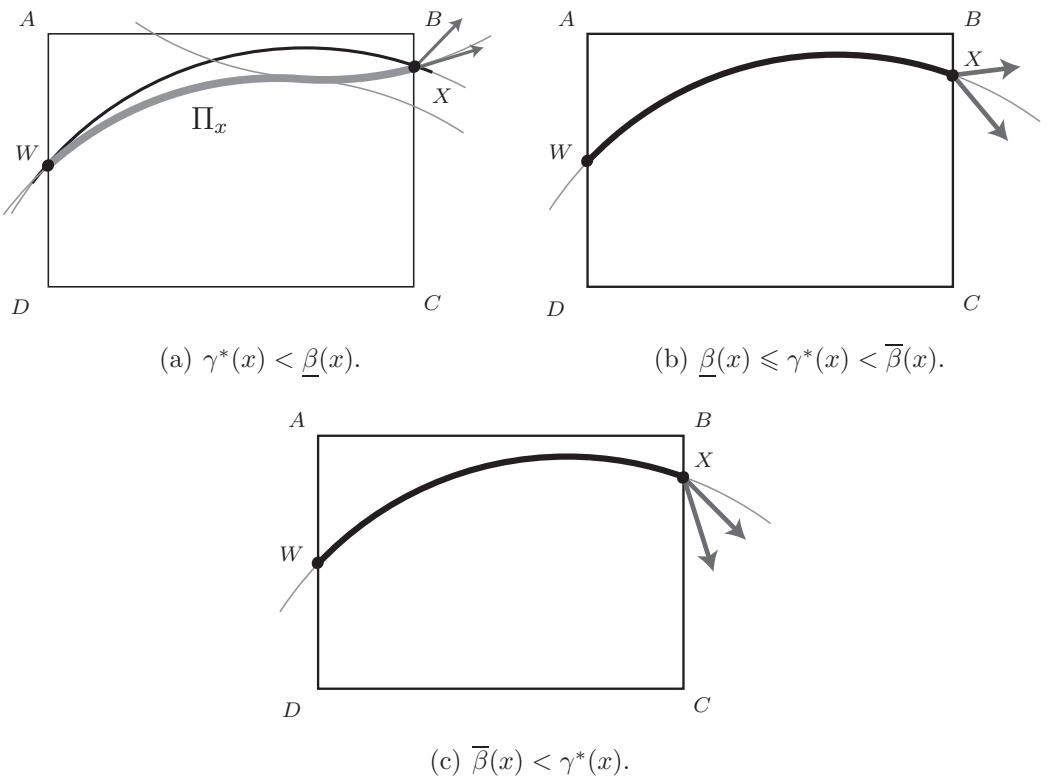
Finally, if $\overline{\beta}(x) < \gamma^*(x)$, then Π_x is not Type 1 admissible, and moreover, it follows by Corollary C.10 that there exists no Type 1 admissible path from W to X (see Fig. C.9(c)).

D. Let $X = (d_1, x)$ be a point on the segment YZ such that $x \in (n_2, m_2] \cap [y, z]$ if $n_2 \leqslant z$ (otherwise we ignore this case). This sub-case is identical to sub-case B of Case 2.

E. Let $X = (d_1, x)$ be a point on the segment YZ such that $x \in (m_2, z]$ if $m_2 \leqslant z$ (otherwise we ignore this case). This sub-case is identical to sub-case C of Case 2.

In summary, we can construct a family of Type 1 admissible paths Π_x such that $\Pi_x \in \mathfrak{D}$, and using identical arguments after a reflection about the x-axis, we can construct a family of paths $\bar{\Pi}_x$ such that $\bar{\Pi}_x \in \mathfrak{D}$. To show that $\Pi_x = \Upsilon_x$ and $\bar{\Pi}_x = \Lambda_x$, we must show that the families Π_x and $\bar{\Pi}_x$ satisfy the properties (P1) through (P5).

Proof of (P1). Similar to the proof of (P1) for Case 2, hence omitted.



(a) $\gamma^*(x) < \underline{\beta}(x)$.

(b) $\underline{\beta}(x) \leqslant \gamma^*(x) < \overline{\beta}(x)$.

(c) $\overline{\beta}(x) < \gamma^*(x)$.

Figure C.9: Illustration of possible cases when a semi-admissible C^+ path exists between W and Y.

Proof of (P2). By construction, the paths Π_x and $\bar{\Pi}_x$ belong to $\mathfrak{D}$. Thus, property (P2) follows as a direct consequence of Lemma C.7.

Proofs of (P4) and (P5). Since the constructions of Π_x (resp. $\bar{\Pi}_x$) for Case 3 are different only if $x \in [n_1, n_2]$ and if the C^+ arc (resp. C^- arc) from W to X is Type 1 admissible, it suffices to prove (P4) and (P5) for this case only. The property (P4) is an immediate consequence of Lemma C.8, and (P5) is an immediate consequence of Corollary C.9.

Proof of (P3). Since the constructions of Π_x (resp. $\bar{\Pi}_x$) for Case 3 are different only if $x \in [n_1, n_2]$ and if the C^+ arc (resp. C^- arc) from W to X is Type 1 admissible, it suffices to prove that the $\Upsilon'_x(W)$ is continuous on the interval $[n_1, n_2]$. By Step 2c) for Case 3, $\Upsilon'_x(W)$ is continuous wherever $\gamma^*(x) < \underline{\beta}(x)$, and it is continuous wherever

$\gamma^*(x) \geqslant \underline{\beta}(x)$. By (C.9), γ^* is continuous. Since $\underline{\beta}$ is also continuous, it suffices to show that when $\gamma^*(x) = \underline{\beta}(x)$, the solution of (C.10) is also a solution to (C.8, i.e., the switch between steps ii) and iii) of Step 2c) does not cause a discontinuity. To see this, substitute $\underline{\beta}(x) = \gamma^*(x)$ in (C.8) and evaluate

$$\Upsilon'_x(W) = 2\tan^{-1}\left(\frac{a \pm \sqrt{a^2+b^2-c^2}}{b+c}\right),$$

where $a := \sin\gamma^*(x) - d_1/r$, $b := \cos\gamma^*(x) + (x-w)/r$, and $c := 1 - (x-w)\cos\gamma^*(x)/r - d_1\sin\gamma^*(x)/r + ((x-w)^2+d_1^2)/2r^2$. Similarly, the solutions of (C.10) can be evaluated as

$$\Upsilon'_x(W) = 2\tan^{-1}\left(\frac{d_1 \pm \sqrt{d_1^2+(x-w)^2 - ((d_1^2+(x-w)^2)/2r)^2}}{d_1 + (d_1^2+(x-w)^2)/2r}\right),$$

and after suitable algebraic manipulations, the two expressions of $\Upsilon'_x(W)$ can be shown to be equal.

C.2 Traversal across Adjacent Edges

The constructions of the paths Υ_x and Λ_x for traversal across parallel edges are broadly similar to those presented in the previous section for the traversal across parallel edges. Hence, we omit the details of these constructions; instead we present directly the computational procedures for determining $\Upsilon'_x(W)$ and $\Lambda'_x(W)$. First, we state some basic facts concerning the geometry of curvature-bounded paths for traversal across adjacent edges.

A fundamental lemma similar to Lemma C.2 is as follows. We define two functions $\alpha_1^* : [0, d_2] \to \left[0, \frac{\pi}{2}\right]$ and $\alpha_2^* : [0, d_2] \to \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ as follows:

$$\alpha_1^*(w) := \begin{cases} \frac{\pi}{2}, & d_2 - w > r, \\ \cos^{-1}\left(1 - \dfrac{d_2-w}{r}\right), & d_2 - w \leqslant r. \end{cases} \tag{C.11}$$

$$\alpha_2^*(w) := \begin{cases} \frac{\pi}{2}, & r \leqslant d_1/2, \\ \sin^{-1}\left(\dfrac{d_1}{r} - 1\right), & r > d_1/2, \quad w \leqslant \sqrt{2d_1r - d_1^2}, \\ \min\left\{2\tan^{-1}\left(d_1 \pm \hat{d}\right)\right\}, & r > d_1/2, \quad w > \sqrt{2d_1r - d_1^2}, \end{cases} \tag{C.12}$$

where $\hat{d} := \sqrt{(d_1^2 + w^2)(4r^2 - 1)}/(d_1^2 + 2rw + w^2)$.

Lemma C.12. *The maximum possible tangent angle at W for any Type 2 admissible path is given by the function $w \mapsto \alpha^*(w)$, defined as*

$$\alpha^*(w) \;\; := \;\; \min\left\{\alpha_1^*(w), \alpha_2^*(w)\right\} \tag{C.13}$$

Proof. Similar to the proof of Lemma C.2, hence omitted. □

Lemma C.13. *If $r \leqslant \dfrac{d_1^2 + w^2}{2w}$, then the minimum possible tangent angle at W for any Type 2 admissible path is given by the function $w \mapsto \alpha_*(w)$, defined as*

$$\alpha_*(w) \;\; := \;\; \begin{cases} \frac{\pi}{2}, & w > r, \\ \cos^{-1}\left(1 - \dfrac{w}{r}\right), & w \leqslant r. \end{cases} \tag{C.14}$$

Proof. Similar to the proof of Lemma C.2, hence omitted. □

Corollary C.14. *Let $X = (d_1, x)$ be any point on segment YZ, and suppose $r \leqslant \dfrac{d_1^2 + (d_1 - x)^2}{2(d_1 - x)}$. If $d_1 - x < r$ and $\overline{\beta}(x) < -\cos^{-1}(1 - (d_1 - x)/r)$, then there exists no Type 2 admissible path between W and X.*

Corollary C.14 is a direct consequence of Lemma C.13 after a reflection about the vertical axis. This Corollary is similar to Corollaries C.4 and C.3, and it provides a preliminary condition on $\overline{\beta}$ and $\underline{\beta}$ for the existence of Type 2 admissible paths between W and X.

As before, we consider for the ease of analysis, different cases of relations between the curvature bound and the dimensions of the rectangle. For computing $\Lambda'_x(W)$, we consider the following cases.

Case 1A: $0 < r < (w/4)$,

Case 2A: $(w/4) \leqslant r < (w/2)$,

Case 3A: $(w/2) \leqslant r$.

Computation of $\Upsilon'_x(W)$ and $\overline{\alpha}$ for Case 1A (Adjacent Edges)

1: $\Upsilon'_x(W) \leftarrow \alpha^*(w), \quad$ for all $x \in [y, z]$
2: $\mathcal{I} \leftarrow [y, z]$
3: $\overline{\alpha} \leftarrow \max_{x \in \mathcal{I}} \{\Upsilon'_x(W)\} = \alpha^*(w)$

Figure C.10: Analysis of traversal across adjacent edges: $0 < r < (w/4)$.

Similarly, for computing $\Upsilon'_x(W)$, we consider the following cases.

Case 1B: $0 < r < (w/3)$,

Case 2B: $(w/3) \leqslant r < w$,

Case 3B: $w \leqslant r$.

The procedures for computing $\Lambda'_x(W)$ and $\Upsilon'_x(W)$ for each of these cases are provided in Figs. C.10–C.15. Note that the procedures for computing $\Lambda'_x(W)$ and $\Upsilon'_x(W)$ provided in Chapter 5 correspond to Cases 3A and 3B.

Computation of $\Upsilon'_x(W)$ and $\overline{\alpha}$ for Case 2A (Adjacent Edges)

if $3r + r\cos\alpha^*(w) < d_2$ **then**

  $\Upsilon'_x(W) \leftarrow \alpha^*(w), \quad$ for all $x \in [y, z]$

$m \leftarrow r\sin(\alpha^*(w)) +$
$$\sqrt{2r^2\left(1 - \cos(\alpha^*(w)) + 0.5\sin^2(\alpha^*(w))\right) + 2wr\left(1 + \cos(\alpha^*(w))\right) - w^2}$$

if $m \leqslant z$ **then**

  $\Upsilon'_x(W) \leftarrow \alpha^*(w)$ for all $x \in [m, z]$

Compute $\beta^*(x)$ for all $x \in [y, m) \cap [y, z]$ as the solution in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ to

$$\begin{aligned}\left(-\sin\alpha^*(w) + \frac{x}{r}\right)\cos(\beta^*(x)) + \left(\cos\alpha^*(w) - \frac{w}{r}\right)\sin(\beta^*(x)) \\ -\frac{w}{r}\cos\alpha^*(w) - \frac{x}{r}\sin\alpha^*(w) + \frac{w^2 + x^2}{2r^2} \;=\; 1\end{aligned} \tag{C.15}$$

if $\beta^*(x) < \underline{\beta}(x)$ **then**

  $\Upsilon'_x(W)$ is the solution in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ to the equation

$$\begin{aligned}\left(\sin\underline{\beta}(x) - \frac{w}{r}\right)\cos(\Upsilon'_x(W)) + \left(-\cos\underline{\beta}(x) - \frac{x}{r}\right)\sin(\Upsilon'_x(W)) \\ +\frac{x}{r}\cos\underline{\beta}(x) - \frac{w}{r}\sin\underline{\beta}(x) + \frac{w^2 + x^2}{2r^2} \;=\; 1\end{aligned} \tag{C.16}$$

else

  $\Upsilon'_x(W) \leftarrow \alpha^*(w)$

$\mathcal{I} \leftarrow [y, z]$

$\overline{\alpha} \leftarrow \max_{x\in\mathcal{I}}\{\Upsilon'_x(W)\}$

Figure C.11: Analysis of traversal across adjacent edges: $(w/4) \leqslant r < (w/2)$.

Computation of $\Upsilon'_x(W)$ and $\overline{\alpha}$ for Case 3A (Adjacent Edges)

1: **if** $r + r\cos(\alpha^*(w)) < w$ **then**
2: Execute the procedure in Fig. C.11
3: **else**
4: Define m as in Line 3 of Fig. C.11
5: $n \leftarrow r\sin(\alpha^*(w)) + \sqrt{r^2\sin^2(\alpha^*(w)) + 2wr\cos(\alpha^*(w)) - w^2}$
6: **if** $m \leqslant z$ **then**
7: $\Upsilon'_x(W) \leftarrow \alpha^*(w), \quad$ for all $x \in [m, z]$
8: For all $x \in (n, m) \cap [y, z]$ execute Lines 6–12 of Fig. C.11
9: Compute $\gamma^*(x)$ for all $x \in [y, n]$ as the solution in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ to

$$x\cos(\gamma^*(x)) - w\sin(\gamma^*(x)) = \frac{w^2 + x^2}{2r} \tag{C.17}$$

10: **if** $\gamma^*(x) < \underline{\beta}(x)$ **then**
11: $\Upsilon'_x(W)$ is the solution in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ to (C.16)
12: **if** $\underline{\beta}(x) \leqslant \gamma^*(x) < \overline{\beta}(x)$ **then**
13: $\Upsilon'_x(W)$ is the solution in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ to

$$w\cos(\gamma^*(x)) + x\sin(\gamma^*(x)) = \frac{w^2 + x^2}{2r} \tag{C.18}$$

14: **else**
15: There exists no Type 2 admissible path between W and X, in particular, Υ_x does not exist
16: $\mathcal{I} \leftarrow$ largest interval in the set $[y, z] \setminus \left\{x \in ((n, m) \cap [y, z]) : \overline{\beta}(x) < \gamma^*(x)\right\}$
17: $\overline{\alpha} \leftarrow \max_{x\in\mathcal{I}}\{\Upsilon'_x(W)\}$

Figure C.12: Analysis of traversal across adjacent edges: $(w/2) \leqslant r$.

Computation of $\Lambda'_x(W)$ and $\underline{\alpha}$ for Case 1B (Adjacent Edges)

1: $\Lambda'_x(W) \leftarrow \alpha_*(w), \quad$ for all $x \in [y, z]$
2: $\mathcal{I} \leftarrow [y, z]$
3: $\underline{\alpha} \leftarrow \min_{x\in\mathcal{I}}\{\Lambda'_x(W)\} = \alpha_*(w)$

Figure C.13: Analysis of traversal across adjacent edges: $0 < r < (w/3)$.

Computation of $\Lambda'_x(W)$ and $\underline{\alpha}$ for Case 2B (Adjacent Edges)

1: $m_1 \leftarrow r - \sqrt{3r^2 + 2wr - w^2}$

2: $m_2 \leftarrow 2r\sqrt{2} - r\sin(\alpha_*(w))$

3: **if** $m_1 \geqslant y$ or $m_2 \leqslant z$ **then**

4: $\quad \Lambda'_x(W) \leftarrow \alpha_*(w), \quad$ for all $x \in [y, m_1] \cup [m_2, z]$

5: Compute $\beta^*(x)$ for all $x \in (m_1, m_2) \cap [y, z]$ as the solution in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ to

$$\begin{aligned} -\left(\sin\alpha_*(w) + \frac{x}{r}\right)\cos(\beta^*(x)) + \left(\cos\alpha_*(w) + \frac{w}{r}\right)\sin(\beta^*(x)) \\ + \frac{w}{r}\cos\alpha_*(w) + \frac{x}{r}\sin\alpha_*(w) + \frac{w^2 + x^2}{2r^2} \;=\; 1 \end{aligned} \tag{C.19}$$

6: **if** $\overline{\beta}(x) < \beta^*(x)$ **then**

7: $\quad \Lambda'_x(W)$ is the solution in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ to

$$\begin{aligned} \left(\sin\underline{\beta}(x) - \frac{w}{r}\right)\cos(\Lambda'_x(W)) + \left(-\cos\underline{\beta}(x) - \frac{x}{r}\right)\sin(\Lambda'_x(W)) \\ + \frac{x}{r}\cos\underline{\beta}(x) - \frac{w}{r}\sin\underline{\beta}(x) + \frac{w^2 + x^2}{2r^2} \;=\; 1 \end{aligned} \tag{C.20}$$

8: **else**

9: $\quad \Lambda'_x(W) \leftarrow \alpha_*(w)$

10: $\mathcal{I} \leftarrow [y, z]$

11: $\underline{\alpha} \leftarrow \min_{x \in \mathcal{I}} \{\Lambda'_x(W)\}$

Figure C.14: Analysis of traversal across adjacent edges: $(w/3) \leqslant r < w$.

Computation of $\Lambda'_x(W)$ and $\underline{\alpha}$ for Case 3B (Adjacent Edges)

1: Define m_1 and m_2 as in Lines 1–2 of Fig. C.14

2: $n_1 \leftarrow r - \sqrt{r^2 - w^2}$

3: $n_2 \leftarrow \sqrt{2rw - w^2}$

4: **if** $m_1 \geqslant y$ or $m_2 \leqslant z$ **then**

5: $\quad \Lambda'_x(W) \leftarrow \alpha_*(w), \quad$ for all $x \in [m, z]$

6: **if** $n_1 \geqslant y$ or $n_2 \leqslant z$ **then**

7: $\quad$ For all $\{(n_1, m_2) \cup (n_1, m_2)\} \cap [y, z]$, execute Lines 5–11 of Fig. C.14

8: Compute $\gamma^*(x)$ for all $x \in [n_1, n_2] \cap [y, z]$ as the solution in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ to

$$-x\cos(\gamma^*(x)) + w\sin(\gamma^*(x)) \;=\; \frac{w^2 + x^2}{2r}. \tag{C.21}$$

9: **if** $\overline{\beta}(x) < \gamma^*(x)$ **then**

10: $\quad \Lambda'_x(W)$ is the solution in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ to (C.20)

11: **if** $\underline{\beta}(x) \leqslant \gamma^*(x) < \overline{\beta}(x)$ **then**

12: $\quad \Lambda'_x(W)$ is the solution in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ to

$$-w\cos(\gamma^*(x)) - x\sin(\gamma^*(x)) \;=\; \frac{w^2 + x^2}{2r} \tag{C.22}$$

13: **else**

14: $\quad$ There exists no Type 2 admissible path between W and X, in particular, Λ_x does not exist

15: $\mathcal{I} \leftarrow$ largest interval in the set $[y, z] \setminus \left\{x \in ([n_1, n_2] \cap [y, z]) : \gamma^*(x) < \underline{\beta}(x)\right\}$

16: $\overline{\alpha} \leftarrow \max_{x \in \mathcal{I}} \{\Upsilon'_x(W)\}$

Figure C.15: Analysis of traversal across adjacent edges: $w \leqslant r$.

References

[1] http://googleblog.blogspot.com/2010/10/what-were-driving-at.html.

[2] Agarwal, P. K. and Wang, H., "Approximation algorithms for curvature-constrained shortest paths," *SIAM Journal on Computing*, vol. 30, no. 6, pp. 1739–1772, 2001.

[3] Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T. A., Ho, P.-H., Nicollin, X., Olivero, A., Sifakis, J., and Yovine, S., "The algorithmic analysis of hybrid systems," *Theoretical Computer Science*, vol. 138, pp. 3 – 34, 1995.

[4] Alur, R., Henzinger, T. A., Lafferriere, G., and Pappas, G. J., "Discrete abstractions of hybrid systems," *Proceedings of the IEEE*, vol. 88, pp. 971–984, July 2000.

[5] Antsaklis, P. J. and Michel, A. N., *A Linear Systems Primer*. Boston, MA: Birkhäuser, 2007.

[6] Athans, M. and Falb, P. L., *Optimal Control*. Dover Publications, Inc., 2007.

[7] Backer, J. and Kirkpatrick, D., "Finding curvature-constrained paths that avoid polygonal obstacles," in *Proceedings of Twenty-third Annual Symposium on Computational Geometry*, (Gyeongju, South Korea), pp. 66–73, June 68, 2007.

[8] Bakolas, E. and Tsiotras, P., "On the generation of nearly optimal, planar paths of bounded curvature and bounded curvature gradient," in *Proceedings of 2009 American Control Conference*, (St. Louis, MO), pp. 385–390, June 2009.

[9] Barraquand, J., Langlois, B., and Latombe, J.-C., "Numerical potential field techniques for robot path planning," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-22, no. 2, pp. 224–241, 1992.

[10] Barraquand, J. and Latombe, J.-C., "Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles," in *Proceedings of 1991 IEEE International Conference on Robotics and Automation*, (Sacramento, CA), pp. 2328 – 2333, April 1991.

[11] Behnke, S., "Local multiresolution path planning," *Lecture Notes in Artificial Intelligence*, vol. 3020, pp. 332–43, 2004.

[12] Belta, C., Bicchi, A., Egerstedt, M., Frazzoli, E., Klavins, E., and Pappas, G. J., "Symbolic planning and control of robot motion," *IEEE Robotics and Automation Magazine*, pp. 61 – 70, March 2007.

[13] Belta, C., Isler, V., and Pappas, G. J., "Discrete abstractions for robot motion planning and control in polygonal environments," *IEEE Transactions on Robotics*, vol. 21, pp. 864 – 874, October 2005.

[14] Bereg, S. and Kirkpatrick, D., "Curvature-bounded traversals of narrow corridors," in *Proceedings of the Twenty-first Annual Symposium on Computational Geometry*, (Pisa, Italy), pp. 278–287, 2005.

[15] Bertsekas, D. P., *Dynamic Programming and Optimal Control*, vol. 1. Athena Scientific, Belmont, MA, 2000.

[16] Bertsekas, D. P. and Rhodes, I. B., "On the minimax reachability of target sets and target tubes," *Automatica*, vol. 7, pp. 233–247, 1971.

[17] Bhattacharya, P. and Gavrilova, M., "Road map-based path planning: Using the Voronoi diagram for a clearance-based shortest path," *IEEE Robotics and Automation Magazine*, vol. 15, pp. 58–66, June 2008.

[18] Boissonnat, J.-D., Ghosh, S. K., Kavitha, T., and Lazard, S., "An algorithm for computing a convex and simple path of bounded curvature in a simple polygon," *Algorithmica*, vol. 34, pp. 109–156, 2002.

[19] Boissonnat, J.-D., Cérézo, A., and Leblond, J., "Shortest paths of bounded curvature in the plane," in *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, (Nice, France), pp. 2315–2320, May 1992.

[20] Branicky, M. S., "Multiple lyapunov functions and other analysis tools for switched and hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 475–482, 1998.

[21] Branicky, M. S., Borkar, V. S., and Mitter, S. K., "A unified framework for hybrid control: Model and optimal control theory," *IEEE Trans. Automatic Control*, vol. 43, pp. 31–45, January 1998.

[22] Brooks, R. A. and Lozano-Pérez, T., "A subdivision algorithm in configuration space for findpath with rotation," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, pp. 224–233, Mar–Apr 1985.

[23] Bui, X.-N., Boissonnat, J.-D., Souères, P., and Laumond, J.-P., "Shortest path synthesis for dubins non-holonomic robot," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, (San Diego, CA, USA), pp. 2–7, May 1994.

[24] Carrioli, L., "Unsupervised path planning of many asynchronously self-moving vehicles," in *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems IROS '91*, pp. 555–559, 1991.

[25] Cassandras, C. G. and Lafortune, S., *Introduction to Discrete Event Systems*. New York, NY: Springer, 2nd ed., 2008.

[26] Chen, D. Z., Szczerba, R. J., and Uhran Jr., J. J., "Framed-quadtree approach for determining euclidean shortest paths in a 2D environment," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 5, pp. 668–681, 1997.

[27] Cherif, M., "Kinodynamic motion planning for all-terrain wheeled vehicles," in *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, (Detroit, MI.), pp. 317 – 322, May 1999.

[28] Cho, J. T. and Nam, B. H., "A study on the fuzzy control navigation and the obstacle avoidance of mobile robot using camera," in *Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Conference*, vol. 4, pp. 2993 – 2997, 2000.

[29] Choset, H., *Sensor Based Motion Planning: The Hierarchical Generalized Voronoi Graph*. PhD thesis, California Institute of Technology, Pasadena, CA., 1996.

[30] Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., and Thrun, S., *Principles of Robot Motion: Theory, Algorithms, and Implementations*. The MIT Press, 2005.

[31] Cipra, B., "Parlez-vous wavelets?," in *What's Happening in the Mathematical Sciences*, vol. 2, American Mathematical Society, 1994.

[32] Conner, D. C., Rizzi, A. A., and Choset, H., "Composition of local potential functions for global robot control and navigation," in *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3546–3550, Las Vegas, Nevada 2003.

[33] Coombs, D., Murphy, K., Lacaze, A., and Legowik, S., "Driving autonomously offroad up to 35 km/h," in *Proceedings of the 2000 International Vehicles Conference*, 2000.

[34] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C., *Introduction to Algorithms*. MIT Press, 2nd ed., 2001.

[35] Cortes, J., "Discontinuous dynamical systems," *IEEE Control Systems Magazine*, vol. 28, pp. 36–73, 2008.

[36] Cunha, S. R., de Matos, A. C., and Pereira, F. L., "An automatic path planning system for autonomous robotic vehicles," in *Proceedings of the IECON '93 International Conference on Industrial Electronics, Control, and Instrumentation*, (Maui, HI, USA), pp. 1442–1447, November 1993.

[37] Daubechies, I., *Ten Lectures on Wavelets*. CBMS-NSF Lecture Notes, 61, SIAM, 1994.

[38] de Berg, M., van Kreveld, M., and Overmars, M., *Computational Geometry: Algorithms and Applications*. Berlin: Springer, 1997.

[39] Donald, B., Xavier, P., Canny, J., and Reif, J., "Kinodynamic motion planning," *Journal of the Association for Computing Machinery*, vol. 40, pp. 1048–1066, November 1993.

[40] Dubins, L. E., "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, pp. 497–516, July 1957.

[41] Elfes, A., "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46 – 57, 1989.

[42] Fainekos, G. E., Girard, A., and Pappas, G. J., "Hierarchical synthesis of hybrid controllers from temproal logic specifications," in *Hybrid Systems: Computation and Control* (Bemporad, A., Bicchi, A., and Buttazzo, G., eds.), LNCS 4416, pp. 203 – 216, 2007.

[43] Fainekos, G. E., Kress-Gazit, H., and Pappas, G. J., "Hybrid controllers for path planning: A temporal logic approach," in *Proceedings of the 44th IEEE Conference on Decision and Control*, (Seville, Spain), pp. 4885 – 4890, 12 – 15 Dec. 2005.

[44] Fortune, S. and Wilfong, G., "Planning constrained motion," *Annals of Mathematics and Artificial Intelligence*, vol. 3, pp. 21–82, 1991.

[45] Fraichard, T. and Scheuer, A., "From reeds and shepp's to continuous curvature paths," *IEEE Transactions on Robotics*, vol. 20, pp. 1025–1035, December 2004.

[46] Frazzoli, E., *Robust Hybrid Control for Autonomous Vehicle Motion Planning*. PhD thesis, Massachusetts Institute of technology, 2001.

[47] Frazzoli, E., Dahleh, M. A., and Feron, E., "Real-time motion planning for agile autonomous vehicles," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 1, pp. 116–129, 2002.

[48] Ge, S. S. and Cui, Y. J., "Dynamic motion planning for mobile robots using potential field method," *Autonomous Robots*, vol. 13, no. 3, pp. 207–222, 2002.

[49] Goebel, R., Sanfelice, R. G., and Teel, A. R., "Hybrid dynamical systems," *IEEE Control Systems Magazine*, vol. 29, no. 2, pp. 28 – 93, 2009.

[50] Haddad, W. M., Chellaboina, V., and Nersesov, S. G., *Impulsive and Hybrid Dynamical Systems: Stability, Dissipativity, and Control*. Princeton University Press, 2006.

[51] Haghverdi, E., Tabuada, P., and Pappas, G. J., "Bisimulation relations for dynamical and control systems," *Electronic Notes in Theoretical Computer Science*, vol. 69, pp. 120–136, 2003.

[52] Hedlund, S. and Rantzer, A., "Optimal control of hybrid systems," in *Proceedings of the 38th IEEE Conference on Decision and Control*, (Phoenix, AZ, USA), pp. 3972–3977, 1999.

[53] Hedlund, S. and Rantzer, A., "Convex dynamic programming for hybrid systems," *IEEE Transactions on Automatic Control*, vol. 47, pp. 1536–1540, 2002.

[54] Holland, J. M., *Designing Autonomous Mobile Robots: Inside the Mind of an Intelligent Machine*. Oxford, UK.: Newnes Press, 2004.

[55] Hourtash, A. and Tarokh, M., "Manipulator path planning by decomposition: Algorithm and analysis," *IEEE Transactions on Robotics and Autonmation*, vol. 17, pp. 842–856, 2001.

[56] Hsu, D., Kindel, R., Latombe, J.-C., and Rock, S., "Randomized kinodynamic motion planning with moving obstacles," *International Journal of Robotics Research*, vol. 21, pp. 233–255, March 2002.

[57] Huang, H.-P. and Chung, S.-Y., "Dynamic visibility graph for path planning," in *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robotics and Systems*, (Sendai, Japan), pp. 2813–2818, Sep. 28 – Oct. 2 2004.

[58] Huckenbeck, U., *Extremal Paths in Graphs*. Berlin, Germany: Akademie Verlag, 1997.

[59] Hwang, J. Y., Kim, J. S., Lim, S. S., and Park, K. H., "A fast path planning by path graph optimization," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 33, pp. 121–127, January 2003.

[60] Hwang, Y. K. and Ahuja, N., "Gross motion planning - a survey," *ACM Computing Surveys*, vol. 24, pp. 219–291, September 1992.

[61] Hwang, Y. K. and Ahuja, N., "A potential field approach to path planning," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 1, pp. 23–32, 1992.

[62] Ilkyun, J., Seewong, J., and Youngouk, K., "Mobile robot navigation using difference of wavelet SIFT," in *Proceedings of the 2009 Second International Conference on Machine Vision*, pp. 286 – 292, 2009.

[63] Jacobs, P. and Canny, J., *Nonholonomic Motion Planning*, ch. Planning Smooth Paths for Mobile Robots, pp. 271–342. Kluwer Acadenic, Norwell, MA, 1992.

[64] Jaillet, L., Cortés, J., and Siméon, T., "Sampling-based path planning on configuration-space costmaps," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 647–659, 2010.

[65] Jaillet, L., Yershova, A., LaValle, S., and Siméon, T., "Adaptive tuning of the sampling domain for dynamic-domain RRTs," in *Proceedings to the IEEE/RSJ Interational Conference on Intelligent Robots and Systems IROS*, (Edmonton, Canada), pp. 4086–4091, Aug 2–6 2005.

[66] Janet, J. A., Luo, R. C., and Kay, M. G., "The essential visibility graph: An approach to global motion planning for autonomous mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Nagoya, Aichi, Japan), 1995.

[67] Jaulin, L., "Path planning using intervals and graphs," *Reliable Computing*, vol. 7, pp. 1–15, 2001.

[68] Jiang, K., Seneviratne, L. D., and Earles, S. W. E., "Finding the 3D shortest path with visibility graph and minimum potential energy," in *Proceedings of the IEEE/RSJ International Conference of Intelligent Robots and Systems IROS '93*, (Tokyo, Japan), July 26 – 30 1993.

[69] Jung, D., *Hierarchical Path Planning and Control of a Small Fixed-wing UAV: Theory and Experimental Validation*. PhD thesis, Georgia Institute of Technology, 2007.

[70] Kalra, N., Ferguson, D., and Stentz, A., "Incremental reconstruction of generalized Voronoi diagrams on grids," *Robotics and Autonomous Systems*, vol. 57, pp. 123–128, Feb 2009.

[71] Kambhampati, S. and Davis, L. S., "Multiresolution path planning for mobile robots," *IEEE Journal of Robotics and Automation*, vol. RA-2, pp. 135–45, September 1986.

[72] Karaman, S. and Frazzoli, E., "Incremental sampling-based optimal motion planning," in *Robotics: Science and Systems*, 2010.

[73] Kavraki, L. E. and Latombe, J.-C., "Randomized preprocessing of configuration space for fast path planning," Tech. Rep. STAN-CS-93-1490, Dept. Computer Science, Stanford University, Stanford, CA, 1993.

[74] Kavraki, L. E., Švestka, P., Latombe, J.-C., and Overmars, M. H., "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[75] Khalil, H. K., *Nonlinear Systems*. Prentice Hall, 3rd ed., 2002.

[76] Khatib, O., "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research*, vol. 5, pp. 90–98, 1986.

[77] Kim, C.-T. and Lee, J.-J., "Mobile robot navigation using multi-resolution electrostatic potential field," in *Proceedings of the 32nd Annual Conference of IEEE Industrial Electronics Society, 2005, IECON 2005*, 2005.

[78] Kim, J.-O. and Khosla, P. K., "Real-time obstacle avoidance using harmonic potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 338–349, 1992.

[79] Kim, S.-W. and Boley, D., "Analytical potential fields and control strategies for motion planning," in *Tasks and Methods in Applied Artificial Intelligence*, vol. 1416 of *LNCS*, pp. 85–94, Springer, 1998.

[80] Kirk, D. E., *Optimal Control Theory – An Introduction*. Dover Publications, Inc., 1998.

[81] Kloetzer, M., *Symoblic Motion Planning and Control*. PhD thesis, Boston University, 2008.

[82] Kloetzer, M. and Belta, C., "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, pp. 287–297, February 2008.

[83] Koenig, S. and Likhachev, M., "Fast replanning for navigation in unknown terrain," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 354–363, 2005.

[84] Koenig, S., Likhachev, M., and Furcy, D., "Lifelong planning A*," *Artificial Intelligence*, vol. 155, pp. 93–146, May 2003.

[85] Koenig, S., Likhachev, M., Liu, Y., and Furcy, D., "Incremental heuristic search in AI," *Artificial Intelligence Magazine*, vol. 25, pp. 99–112, 2004.

[86] Kreyszig, E., *Differential Geometry:*. New York: Dover Publications, Inc., 1991.

[87] Krozel, J. and Andrisani, D., "Navigation path planning for autonomous aircraft: Voronoi diagram approach," *Journal of Guidance, Control, and Dynamics*, vol. 13, no. 6, pp. 1152–154, 1990.

[88] Kuwata, Y. and How, J. P., "Stable trajectory design for highly constrained environments using receding horizon control," in *Proceedings of the 2004 American Control Conference*, (Boston, MA), pp. 902 – 907, June 30 – July 2 2004.

[89] Lamiraux, F. and Laumond, J.-P., "Smooth motion planning for car-like vehicles," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 4, pp. 498–502, 2001.

[90] Latombe, J.-C., *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[91] Laumond, J.-P., ed., *Robot Motion Planning and Control*. No. 229 in Lecture Notes in Control and Information Sciences, Springer, 1998.

[92] Laumond, J.-P., Taix, M., Jacobs, P., and Murray, R. M., "A motion planner for nonholonomic mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 5, pp. 577–593, 1994.

[93] LaValle, S. M., *Planning Algorithms*. Cambridge University Press, 2006.

[94] LaValle, S. M. and Kuffner, Jr., J. J., "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, pp. 378–400, May 2001.

[95] LaValle, S. M. and Kuffner, Jr., J. J., "Rapidly-exploring random trees: Progress and prospects," in *New Directions in Algorithmic and Computational Robotics* (Donald, B. R., Lynch, K., and Rus, D., eds.), pp. 293–308, AK Peters, 2001.

[96] Lepetič, M., Klancčar, G., Škrjanc, I., Matko, D., and Potočnik, B., "Time optimal path planning considering acceleration limits," *Robotics and Autonomous Systems*, vol. 45, pp. 199–210, 2003.

[97] Liberzon, D., *Switching in Systems and Control*. Boston, MA: Birkhäuser, 2003.

[98] Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., and Thrun, S., "Anytime search in dynamic graphs," *Artificial Intelligence*, vol. 172, no. 14, pp. 1613–1643, 2008.

[99] Lindemann, S. R., Hussein, I. I., and LaValle, S. M., "Real time feedback control for non-holonomic mobile robots with obstacles," in *Proceedings of the 45th IEEE Conference on Decision and Control*, (San Diego, CA, USA), pp. 2406–2411, December 2006.

[100] Lindemann, S. R. and LaValle, S. M., "Smooth feedback for car-like vehicles in polygonal environments," in *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, (Roma, Italy), pp. 3104–3109, April 2007.

[101] Lindemann, S. R. and LaValle, S. M., "Simple and efficient algorithms for computing smooth, collision-free feedback laws over given cell decompositions," *International Journal of Robotics Research*, vol. 28, pp. 600–621, May 2009.

[102] Liu, Y. and Arimoto, S., "Path planning using a tangent graph for mobile robots among polygonal and curved obstacles," *International Journal of Robotics Research*, vol. 11, no. 4, pp. 376–382, 1992.

[103] Lozano-Pérez, T., "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, pp. 560–570, October 1979.

[104] Lui, W. L. D. and Jarvis, R., "A pure vision-based approach to topological SLAM," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Taipei, Taiwan), pp. 3784 – 3791, October 18 – 22 2010.

[105] Maciejowski, J. M., *Predictive Control with Constraints*. Pearson Education Inc., 2002.

[106] Mallat, S. G., "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–93, 1989.

[107] Mesarović, M. D., Macko, D., and Takahara, Y., *Theory of Hierarchical, Multilevel Systems*. Academic Press, New York, 1970.

[108] Mettler, B. and Bachelder, E., "Combining on- and offline optimization techniques for efficient autonomous vehicle's trajectory planning," in *Collection of Technical Papers - AIAA Guidance, Navigation, and Control Conference 2005*, vol. 1 of *499–511*, 2005.

[109] Mirtich, B. and Canny, J., "Using skeletons for nonholonomic path planning among obstacles," in *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, (Nice, France), pp. 2533–2540, May 1992.

[110] Nagatani, K., Iwai, Y., and Tanaka, Y., "Sensor-based navigation for car-like mobile robots based on a generalized Voronoi graph," *Advanced Robotics*, vol. 17, no. 5, pp. 385–401, 2003.

[111] Narayanaswami, R. and Pang, J., "Multiresolution analysis as an approach for tool path planning in nc machining," *Computer-Aided Design*, vol. 35, pp. 167–78, 2003.

[112] Nilsson, N. J., *Artificial Intelligence: A New Synthesis*. San Francisco, CA: Morgan Kauffman Publishers, Inc., 1998.

[113] Noborio, H., Naniwa, T., and Arimoto, S., "A quadtree-based path planning algorithm for a mobile robot," *Journal of Robotic Systems*, vol. 7, no. 4, pp. 555–74, 1990.

[114] Pai, D. K. and Reissell, L.-M., "Multiresolution rough terrain motion planning," *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 19–33, February 1998.

[115] Pappas, G. J., "Bisimilar linear systems," *Automatica*, vol. 39, pp. 2035–2047, 2003.

[116] Paulson, C., Ezekiel, S., and Wu, D., "Wavelet-based image registration," in *Evolutionary and Bio-Inspired Computation: Theory and Applications IV, Proceedings of the SPIE* (O'Donnel, T. H., Blowers, M., and Priddy, K., eds.), vol. 7704, 2010.

[117] Piccoli, B., "Hybrid systems and optimal control," in *Proceedings of the 37th IEEE Conference on Decision and Control*, (Tampa, FL, USA), Dec. 1998.

[118] Plaku, E., Kavraki, L. E., and Vardi, M. Y., "Motion planning with dynamics by a synergistic combination of layers of planning," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 469–482, 2010.

[119] Prazenica, R. J., Kurdila, A. J., Sharpley, R. C., and Evers, J., "Multiresolution and adaptive path planning for maneuver of micro-air-vehicles in urban environments," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (San Francisco, CA), pp. 1–12, 2005.

[120] Ramalingam, G. and Reps, T., "An incremental algorithm for a generalization of the shortest-path problem," *Journal of Algorithms*, vol. 21, pp. 267–305, 1996.

[121] Rao, R. M. and Bopardikar, A. S., *Wavelet Transforms - Introduction to Theory and Applications*. Addison-Wesley, 1998.

[122] Reeds, J. A. and Shepp, L. A., "Optimal paths for a car that goes both forwards and backwards," *Pacific Journal of Mathematics*, vol. 145, no. 2, pp. 367–393, 1990.

[123] Reif, J. and Wang, H., "The complexity of the two dimensional curvature-constrained shortest-path problem," in *Proceedings of the 3rd Workshop on the Algorithmic Foundations of Robotics*, pp. 49–58, 1998.

[124] Rimon, E. and Koditschek, D. E., "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, 1992.

[125] Rippel, E., Bar-Gill, A., and Shimkin, N., "Fast graph-search algorithms for general aviation flight trajectory generation," *Journal of Guidance, Control, and Dynamics*, vol. 28, pp. 801–811, July-August 2005.

[126] Rohnert, H., "Shortest path in the plane with polygonal obstacles," *Information Processing Letters*, vol. 23, pp. 71–76, 1986.

[127] Roque, W. L. and Doering, D., "Trajectory planning for lab robots based on global vision and Voronoi roadmaps," *Robotica*, vol. 23, no. 4, pp. 467–477, 2005.

[128] Rosiglioni, A. and Simina, M., "Kinodynamic motion planning," in *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, vol. 3, pp. 2243–2248, 2003.

[129] Rouchon, P., Fliess, M., Levine, J., and Martin, P., "Flatness, motion planning, and trailer systems," in *Proceedings of the 32nd IEEE Conference on Decision and Control*, (San Antonio, TX, USA), pp. 2700–2705, Dec. 12 – 15 1993.

[130] Russell, S. and Norvig, P., *Artificial Intelligence: A Modern Approach*. NJ, USA: Pearson Education Inc., 2003.

[131] Samet, H., "The quadtree and related hierarchical data structures," *Computing Surveys*, vol. 16, pp. 187–260, June 1984.

[132] Scheuer, A. and Fraichard, T., "Continuous-curvature path planning for car-like vehicles," in *Proceedings of th 1997 IEEE/RSJ International Conference on Intelligent Robots and Systems IROS '97*, (Grenoble, France), pp. 997–1003, September 1997.

[133] Schwartz, J. T. and Sharir, M., *On the Piano Movers' Problem : I. The Case of a Two-Dimensional Rigid Polygonal Body Moving Amidst Polygonal Barriers*, ch. 1, pp. 1–50. Ablex Series in Artificial Intelligence, Norwood, NJ: Ablex Publishing Corporation, 1987.

[134] Schwartz, J. T. and Sharir, M., *On the Piano Movers' Problem : II. General Techniques for Computing Topological Properties of Real Algebraic Manifolds*, ch. 2, pp. 51–96. Ablex Series in Artificial Intelligence, Norwood, NJ: Ablex Publishing Corporation, 1987.

[135] Shiller, Z., "On singular time-optimal control along specified paths," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 4, pp. 561–566, 1994.

[136] Shiller, Z. and Gwo, Y.-R., "Dynamic motion planning of autonomous vehicles," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 2, pp. 241–249, 1991.

[137] Shiller, Z. and Lu, H.-H., "Computation of path constrained time-optimal motions with dynamic singularities," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 114, pp. 34–40, 1992.

[138] Shim, M., Kurtz, J., and Laine, A., "Multi-resolution stereo algorithm via wavelet representations for autonomous navigation," in *Proceedings of the SPIE*, vol. 3723, (Orlando, FL), pp. 319 – 328, April 1999.

[139] Siegwart, R. and Nourbakhsh, I. R., *Introduction to Autonomous Mobile Robots*. Cambridge, MA, USA.: MIT Press, 2004.

[140] Sinopoli, B., Micheli, M., Donato, G., and Koo, T. J., "Vision based navigation for an unmanned aerial vehicle," in *Proceedings of the 2001 IEEE Conference on Robotics and Automation*, pp. 1757–64, 2001.

[141] Souères, P. and Laumond, J.-P., "Shortest path synthesis for a car-like robot," *IEEE Transactions on Automatic Control*, vol. 41, pp. 672–688, May 1996.

[142] Stentz, A., "Optimal and efficient path planning for unknown and dynamic environments," Tech. Rep. CMU-RI-TR-93-20, Carnegie Mellon Robotics Institute, 1993.

[143] Stentz, A., "Optimal and efficient path planning for partially-known environments," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3310 – 3317, 1994.

[144] Sussmann, H. and Tang, G., "Shortest paths for the reeds-shepp car: A worked out example of the use of geometric techniques in nonlinear optimal control," tech. rep., Rutgers Center for Systems and Control (SYCON), September 1991.

[145] Sussmann, H., "A maximum principle for hybrid optimal control problems," in *Proc. 38th IEEE Conf. Decision & Control*, (Phoenix, AZ, USA), 1999.

[146] Tabuada, P., "Controller synthesis for bisimulation equivalence," *Systems and Control Lett.*, vol. 57, pp. 443–452, June 2008.

[147] Tsiotras, P. and Bakolas, E., "A hierarchical on-line path planning scheme using wavelets," in *Proceedings of the European Control Conference*, (Kos, Greece), pp. 2806–2812, July 2–5 2007.

[148] Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M. N., Dolan, J., Duggins, D., Galatali, T., Geyer, C., Gittleman, M., Harbaugh, S., Hebert, M., Howard, T. M., Kolski, S., Kelly, A., Likhachev, M., McNaughton, M., Miller, N., Peterson, K., Pilnick, B., Rajkumar, R., Rybski, P., Salesky, B., Seo, Y.-W., Singh, S., Snider, J., Stentz, A., Whittaker, W., Wolkowicki, Z., Ziglar, J., Bae, H., Brown, T., Demitrish, D., Litkouhi, B., Nickolaou, J., Sadekar, V., Zhang, W., Struble, J., Taylor, M., Darms, M., and Ferguson, D., "Autonomous driving in urban environments: Boss and the Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[149] van der Schaft, A. J., "Equivalence of dynamical systems by bisimulation," *IEEE Transactions on Automatic Control*, vol. 49, pp. 2160–2172, 2004.

[150] van der Schaft, A. J. and Schumacher, H., *An Introduction to Hybrid Dynamical Systems*. Springer, 2000.

[151] Velenis, E. and Tsiotras, P., "Minimum-time travel for a vehicle with acceleration limits: Theoretical analysis and receding horizon implementation," *Journal of Optimization Theory and Applications*, vol. 138, no. 2, pp. 275–296, 2008.

[152] Verwer, B. J. H., "A multiresolution workspace, multiresolution configuration space approach to solve the path planning problem," in *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, pp. 2107–12, 1990.

[153] Švestka, P., "A probabilistic approach to motion planning for car-like robots," Tech. Rep. RUU-CS-1993-18, Dept. Computer Science, Utrecht University, Utrecht, The Netherlands, 1993.

[154] Warren, C. W., "Global path planning using artificial potential fields," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 316–321, 1989.

[155] Wei, L. and Fwa, T. F., "Characterizing road roughness by wavelet transform," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1869, pp. 152 – 158, 2004.

[156] Wiesemann, T., Schiefele, J., and Bader, J., "Multi-resolution terrain depiction and airport navigation function on an embedded SVS," in *Enhanced and Synthetic Vision 2002, Proceedings of the SPIE* (Verly, J. G., ed.), vol. 4713, pp. 106 – 117, 2002.

[157] Winter, S., "Modeling costs of turns in route planning," *GeoInformatica*, vol. 6, no. 4, pp. 345 – 361, 2002.

[158] Wongpiromsarn, T., *Formal Methods for Design and Verification of Embedded Control Systems: Application to an Autonomous Vehicle*. PhD thesis, California Institute of Technology, 2010.

[159] Yang, K. and Sukkarieh, S., "Real-time continuous curvature path planning of UAVs in cluttered environments," in *Proceedings of the 5^{t}h International Symposium on Mechatronics and its Applications*, pp. 1–6, May 2008.

[160] Yang, X. and Meng, M., "Real-time motion planning of car-like robots," in *Proceedings of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems IROS '99*, (Kyongju, Korea), pp. 1298–1303, 1999.

[161] Ye, H., Michel, A. N., and Hou, L., "Stability theory for hybrid dynamical systems," *IEEE Transactions on Automatic Control*, vol. 43, pp. 461–474, 1998.

[162] Yershova, A., Jaillet, L., Siméon, T., and LaValle, S., "Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Barcelona, Spain), pp. 2867–6872, Apr 18–22 2005.

[163] Yguel, M., Aycard, O., and Laugier, C., "Wavelet occupancy grids: A method for compact map building," in *Field and Service Robotics, STAR 25* (Corke, P. and Sukkarieh, S., eds.), pp. 219 – 230, Springer-Verlag, 2006.

[164] Zhu, D. and Latombe, J.-C., "New heuristic algorithms for efficient hierarchical path planning," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 1, pp. 9–20, 1991.

Vita

Raghvendra Cowlagi obtained a Bachelor's degree in Electronics Engineering from the University of Mumbai, India, in 2003; and a Master's degree in Aerospace Engineering from the Indian Institute of Technology Bombay, Mumbai, India, in 2005. Raghvendra worked for a year as Project Engineer at the Department of Aerospace Engineering, Indian Institute of Technology Bombay, following which he joined the Ph. D. program at the School of Aerospace Engineering at Georgia Tech in 2006. The highlights of his academic career include the Student Best Paper Award at the 2009 American Control Conference and the 2005 Aeronautical Society of India Award. His research interests include autonomous mobile vehicles, motion planning, hybrid control, and the safety of large scale systems.